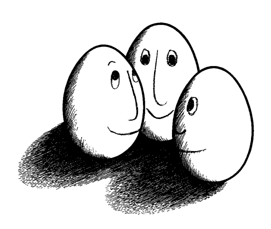
Diplomarbeit

mAGENTa
Ein Internet-Agent
zur automatischen Generierung
von graphischen Benutzeroberflächen
mittels maschineller Lernverfahren
am Beispiel radiologischer Befunde

Kai Bullerdick

Oliver Ritthoff



Diplomarbeit

am Lehrstuhl 8 des Fachbereichs Informatik
der Universität Dortmund

in Zusammenarbeit mit

dem Lehrstuhl für Radiologie und MikroTherapie
der Universität Witten/Herdecke

6. Oktober 1999

Betreuer:
Prof. Dr. Katharina Morik
Prof. Dr. med. Dietrich H.W. Grönemeyer

1 Vorwort

An dieser Stelle möchten wir all denjenigen danken, die einen Beitrag zu dieser Diplomarbeit geleistet haben.

Für die Betreuung der Diplomarbeit möchten wir uns bei Prof. Dr. Katharina Morik und Prof. Dr. med. Dietrich H.-W. Grönemeyer bedanken.

Besonderer Dank gilt Dr. med. Paul Kriener für die fachliche Beratung und Unterstützung auf radiologischem Gebiet, sowie für die Bereitstellung und Bearbeitung der Befunddaten. Ebenso bedanken wir uns bei Dipl.-Inform. Jörg Holstein für seine spontane und unermüdliche Hilfe in allen Belangen.

Herzlicher Dank gebührt Dipl.-Inform. Thorsten Joachims für seine Beratung in Fragen des maschinellen Lernens sowie für zahlreiche Kommentare und Hinweise bezüglich der Gestaltung der Diplomarbeit. Ein besonderer Dank geht im übrigen an die Mitarbeiter des Lehrstuhls VIII des Fachbereiches Informatik für anregende Diskussionen nicht nur auf dem Gebiet der Künstlichen Intelligenz.

Dank geht ebenfalls an Michael Kleinwegen für die gewissenhafte und unermüdliche Durchsicht der Diplomarbeit.

Ganz speziellen Dank an unsere Eltern sowie an Annette und Sonja für ihre Geduld und Unterstützung, ohne die diese Arbeit nicht möglich gewesen wäre.

Die vorgestellte Arbeit ist in enger Zusammenarbeit der beiden Autoren entstanden. Dennoch können die nachfolgenden Kapitel wie folgt den einzelnen Autoren zugeordnet werden:

Kai Bullerdick:

Kapitel 4 Intelligente Agenten,

Kapitel 7 Verarbeitung der Befundtexte,

Kapitel 9 TokenComponentInterface,

Kapitel 10 Befunddokumentation

Kapitel 11 Benutzerhandbuch

Oliver Ritthoff:

Kapitel 3 Radiologie,

Kapitel 5 Maschinelle Lernverfahren,

Kapitel 6 Systemarchitektur,

Kapitel 8 DFA-Generierung,

Kapitel 12 Testergebnisse

Inhaltsverzeichnis

I Grundlagen

2 Einleitung

2.1 Abstract

Die Erfassung von gleichartigen Texten ist eine in vielen Bereichen immer wieder vorkommende Aufgabe. Besonders in der Medizin ist das Erstellen von Befundtexten eine Routineaufgabe, die von Ärzten (meist unter Zeitdruck) durchgeführt wird. Der hier vorgestellte Softwareagent unterstützt den Arzt aktiv bei seiner Arbeit, indem er ihn bei der Erfassung von Befunden beobachtet, aus seiner Vorgehensweise lernt, und eine graphische Benutzerschnittstelle generiert. Diese Benutzerschnittstelle leitet den Arzt anschließend strukturiert durch weitere Befundungsvorgänge und ermöglicht durch bloßes "Anklicken" von Oberflächenelementen die effiziente Erstellung von Befundtexten.

2.2 Aufgabenstellung

Die Befundung von Patienten gehört zu den täglichen Aufgaben eines Arztes, sowohl in der Praxis als auch in der Klinik. Obwohl die Befunderfassung eine Routinearbeit ist, stellt sich jeder Patient als individueller Fall mit speziellen Eigenheiten dar. Der Einsatz eines Befundungsunterstützungssystems könnte hier den Arzt bei dieser Routinearbeit entlasten und es ihm so ermöglichen, sich mehr auf seine diagnostische und therapeutische Tätigkeit zu konzentrieren.

Ein solches System sollte für eine sinnvolle Anwendung verschiedene Anforderungen erfüllen, die im folgenden erläutert werden:

Anforderungen an das System	
1. Kategorisierung	4. Strukturierung
2. Vereinheitlichung	5. Fehlervermeidung
3. Adaptivität	6. kooperative Nutzung

Tabelle 1 Anforderungen an das System

1. Kategorisierung

Ein Arzt muß immer wieder die gleichen Krankheitsbilder bzw. Organsysteme befunden. Besonders Fachärzte haben hier ein eng umfaßtes Aufgabengebiet, welches immer wieder zu ähnlichen Befunden führt. Oft werden auch Befunde speziell für eine Diagnose mit einem Untersuchungsgerät erstellt. Hier wären insbesondere die radiologischen Untersuchungen zu nennen, die beispielhaft für eine Anwendung des Systems genutzt worden sind. Jede Untersuchungsart wird speziell für ein Krankheitsbild bzw. eine Organregion genutzt. Hierdurch ergibt sich eine Kategorisierung der Befunddomäne.

2. Strukturierung

Im Idealfall sollte ein Befund gut strukturiert sein. Er sollte einem bestimmten Schema folgen, das als Richtlinie für eine Befundkategorie (z. B. Computertomogramme der Halswirbelsäule) dient. Es müssen bei der Befundung alle für die Diagnose wichtigen Sachverhalte erkannt und so präzise wie möglich dargelegt werden. Alle auf den Untersuchungsaufnahmen sichtbaren Organe sollten befundet werden. Falls einzelne Organe sich ohne pathologischen Befund darstellen sollte dies entsprechend festgehalten werden. Ferner sollten die diagnostisch wichtigen Erkenntnisse vor den weniger wichtigen genannt werden.

3. Vereinheitlichung

Viele Ärzte führen ähnliche Untersuchungen durch. Trotzdem haben diese Befunde oftmals sehr unterschiedliches Aussehen. Dies hängt u.a. damit zusammen, daß jeder Arzt seinen individuellen Befundungsstil hat. Außerdem werden in der medizinischen Fachsprache sehr oft Synonyme, Akronyme und Abkürzungen in beliebiger Kombination verwendet. Für Fachworte gibt es mitunter verschiedene gebräuchliche Schreibweisen, die selbst innerhalb eines Befundes zu finden sind. Eine knappe aber präzise Formulierung ist einer zu ausführlichen vorzuziehen. Eine Vereinheitlichung dieser Punkte ist für gute Befunde erstrebenswert, da dies die Lesbarkeit erhöht und einen uniformen Befundungsstil für eine Klinik oder Praxis bietet.

4. Fehlervermeidung

Das System sollte bei einem neu eingegebenen Befund automatisch feststellen können, daß ein Wort oder ein Satz des neuen Befundes schon einmal als falsch deklariert und korrigiert worden ist. Es sollte dies selbständig feststellen und ohne weitere Anweisung korrigieren. Oftmals werden die Befunddaten vom Arzt zum Teil handschriftlich oder mittels Diktat erfaßt. Nachträglich müssen sie zur Weiterverarbeitung im Rechner von einer Schreibkraft in eine elektronische Form übertragen werden. Es ist zu beobachten, daß in diesem Medienbruch nicht nur eine Fehlerquelle liegt, sondern dies auch zeitlich

von Nachteil ist. Bei der manuellen Eingabe von Texten über eine Tastatur ist es oft der Fall, daß sich einfache Schreibfehler, wie z.B. Buchstabendreher, fehlende Buchstaben usw. einschleichen.

5. Adaptivität

Das System sollte an die Belange des Benutzers angepaßt werden können. Darüber hinaus ist es wünschenswert, durch die selbständige Anpassung eine erhöhte Nutzbarkeit zu erreichen. Es muß einfach möglich sein, Informationen zu entfernen, zu ändern oder zu ergänzen. Ist z.B. ein Wort falsch geschrieben, so sollte dieses durch die korrekte Schreibweise ersetzt werden können, bzw. automatisch ersetzt werden.

6. kooperative Nutzung

Jeder Arzt befundet nur auf Grundlage seines eigenen Kenntnisstandes, ohne während der Befundung auf die Erfahrungen anderer zurückgreifen zu können. In vielen Kliniken und großen Praxen sind untereinander vernetzte Systeme zu finden. Die Infrastruktur, um das Potential der vorhandenen Ärzte im Intra- oder Internet kooperativ zu nutzen, ist gegeben, wird aber noch nicht ausgeschöpft. Da die eingesetzten Netzwerke oftmals heterogen sind, ist ein plattformunabhängiges System sinnvoll. Durch die gemeinsame Nutzung des gleichen Systems können Fehler leichter aufgespürt werden, die ein einzelner übersähe.

7. Domänenunabhängigkeit

Prinzipiell soll das System unabhängig sein von der Wahl der jeweiligen Kategorie. Hiermit ist nicht nur gemeint, daß beliebige Befundkategorien unterstützt werden sollen sondern auch gänzlich andere Arten strukturierter Texte wie z. B. Formbriefe aus behördlichen Einrichtungen. Der Rahmen in dem die Evaluation des Systems erfolgt, soll jedoch beschränkt bleiben auf einzelne radiologische Domänen.

2.3 Bisherige Lösungsansätze

Bestehende Befundungsunterstützungssysteme haben das Manko, daß sie von den Ärzten als nicht sehr benutzerfreundlich empfunden werden. Bei Textbaustein-basierten Systemen werden die Befundtexte aus vorgefertigten Sätzen zusammengefügt. Diese Sätze haben aber keinen Bezug untereinander, so daß der Benutzer nicht strukturiert durch den Befundungsvorgang geleitet wird. Auch ist eine Änderung der vorgefertigten Sätze nicht immer einfach möglich, sondern kann erst im schon erstellten Befund vorgenommen werden. Ebenso unflexibel stellen sich wissensbasierte Systeme dar. Während eine strukturierte Befundungsanleitung möglich ist, ist es hier ebenso schwierig für einen Arzt das "Hintergrundwissen" dieser Systeme den eigenen

Bedürfnissen entsprechend zu modifizieren. Für jede Befundkategorie muß außerdem eine neue Wissenbasis aufwendig erstellt werden.

2.4 Lösungsansatz durch einen Softwareagenten

Das hier vorgestellte System soll den Arzt aktiv bei der Befunderstellung unterstützen, indem es das Erfassen von Befunden beobachtet, aus den erfaßten Befunden lernt, und ihn bei seinen nächsten Befundungen strukturiert anleitet.

Dieser adaptive Softwareagent konstruiert also aus den gelernten Befunden eine leicht zu bedienende graphische Benutzeroberfläche, die es dem befundenden Arzt ermöglicht, durch bloßes "Anklicken" einzelner Oberflächenelemente effizient einen Befund zu verfassen. Während der Benutzung des Agenten paßt dieser sich selbständig an, um für zukünftige Befundungen eine bessere Nutzbarkeit zu erreichen, und den Arzt strukturiert durch seinen Befundungsvorgang zu leiten.

Technisch gesprochen soll das System Befundtexte einlesen, mittels maschineller Lernverfahren eine interne Repräsentation davon erstellen, und daraus eine graphische Oberfläche erzeugen, mit der weitere Befunde erfaßt werden können.

Für jede Befundungskategorie (z.B. EBT des Herzens, oder CT der HWS) ist es möglich, jeweils einen Agenten zu trainieren. Dies kann auch von verschiedenen Ärzten getrennt geschehen. Die Agenten können jedoch ihre einzeln akkumulierten "Erfahrungen" zusammenfassen, so daß sie jeweils bei weiteren Befundungen ein umfangreicheres Hintergrundwissen zur Verfügung haben.

Da die befundenden Ärzte i.A. nicht alle einen einzigen zentralen Computer zur Befunderfassung nutzen, sondern eher räumlich getrennt über ein Netzwerk (Intranet oder auch Internet) verbunden arbeiten, ist es notwendig, daß die Kooperation zwischen ihnen auch über diese Kanäle erfolgt. Um in heterogenen Netzwerken nicht für jede Betriebssystem-Plattform einen eigenen Agenten zu implementieren, ist es sinnvoll, dies in einer plattformunabhängigen Programmiersprache zu tun. Für größtmögliche Unabhängigkeit und Flexibilität wurde hier Java 2 gewählt, um mit den Vorzügen von Swing eine vom Betriebssystem unabhängige Darstellung zu erreichen.

Einmal gelernte Befundkategorien können von allen Ärzten genutzt werden, auch wenn der einzelne noch keine Befunde für eine spezielle Kategorie verfaßt hat. Auch können so die Ergebnisse von Befundungsvorgängen verschiedener Ärzte zusammengefügt werden, um zu einem umfassenderen System zu gelangen. Wenn ein Agent noch keine "Erfahrung" über eine bestimmte Eigenart eines Patienten hat, so ist es möglich, daß dieser auf die eines anderen zurückgreift und beide miteinander kombiniert.

Außerdem wird der Anteil der Fehler (z. B. Rechtschreibfehler) durch die kooperative Nutzung des Systems im Rahmen der Befunderstellung minimiert. Sollten einzelne Ärzte entsprechende Fehler feststellen, können sie diese umgehend korrigieren und die verbesserte Version den restlichen Benutzern zugänglich machen.

2.5 Aufbau der Diplomarbeit

Die Diplomarbeit läßt sich in drei große Abschnitte einteilen:

Die Abschnitte der Diplomarbeit
1. Die Grundlagen
2. Die Realisierung
3. Die Auswertung

Tabelle 2 Die Abschnitte der Diplomarbeit

Die Grundlagen und theoretischen Hintergründe werden in den Kapiteln 3, 4 und 5 behandelt. Nach der Einführung in die radiologische Domäne im Allgemeinen und die Befunderhebung im Speziellen in Kapitel 3, befaßt sich Kapitel 4 mit dem Begriff der intelligenten Softwareagenten. Kapitel 5 gibt einen Überblick über maschinelle Lernverfahren, die häufig die Basis intelligenter Systeme darstellen.

Der zweite Abschnitt legt dar, wie die beschriebenen Grundlagen unter Berücksichtigung der Aufgabenstellung in ein funktionierendes System umgesetzt worden sind. In Kapitel 6 wird ein Überlick über den Aufbau des erstellten Systems gegeben. Kapitel 7 zeigt, wie die Befundungstexte vorverarbeitet werden müssen, damit in Kapitel 8 der Lernalgorithmus darauf arbeiten kann. Anhand dieser Ergebnisse wird im folgenden Kapitel 9 beschrieben, wie daraus eine Benutzeroberfläche generiert werden kann. Kapitel 10 gibt darüber Auskunft, wie durch die Eingaben in die Benutzeroberfläche nun wieder Befundungstexte erstellt werden. Kapitel 11 beinhaltet ein ausführliches und umfangreiches Benutzerhandbuch.

Der letze Abschnitt behandelt die Auswertung des erstellten Systems. Kapitel 12 zeigt vergleichende Testergebnisse anhand von drei verschiedenen Domänen. Das 13. Kapitel liefert eine abschließende Betrachtung der Diplomarbeit hinsichtlich des erzielten Ergebnisses.

3 Radiologie

Dieses Kapitel beginnt mit einem Einblick in die wichtigsten bildgebenden radiologischen Untersuchungsmethoden. Dabei liegt der Schwerpunkt der Ausführungen primär auf den in der Diplomarbeit relevanten Methoden der Computertomographie (CT), der Elektronenstrahltomographie (EBT) und der Magnetresonanztomographie (MRT).

Im zweiten Abschnitt dieses Kapitels wird der radiologische Befundungsvorgang diskutiert. Dabei wird neben den sprachlichen Besonderheiten der radiologischen Fachsprache auch die Struktur eines radiologischen Befundes untersucht. Anschließend werden die Anwendungsmöglichkeiten des erstellten Systems im Rahmen der radiologischen Ausbildung sowie zur Unterstützung des Arztes beim Befundungsvorgang diskutiert.

Der letzte Abschnitt behandelt die generellen Vorteile von Systemen zur Befundungsunterstützung im Vergleich zu herkömmlichen Befundungsverfahren und gibt einen Einblick in die Methoden von Textbaustein- bzw. wissensbasierten Systemen.

3.1 Bildgebende Verfahren in der Radiologie

Die folgende Abbildung gibt einen Überblick über die wichtigsten radiologischen Untersuchungsverfahren (siehe hierzu auch [Seelos et al. 1997]). Die grau unterlegten Verfahren (Computertomographie, Magnetresonanztomographie und Elektronenstrahltomograhpie) werden in diesem Kapitel ausführlicher behandelt.

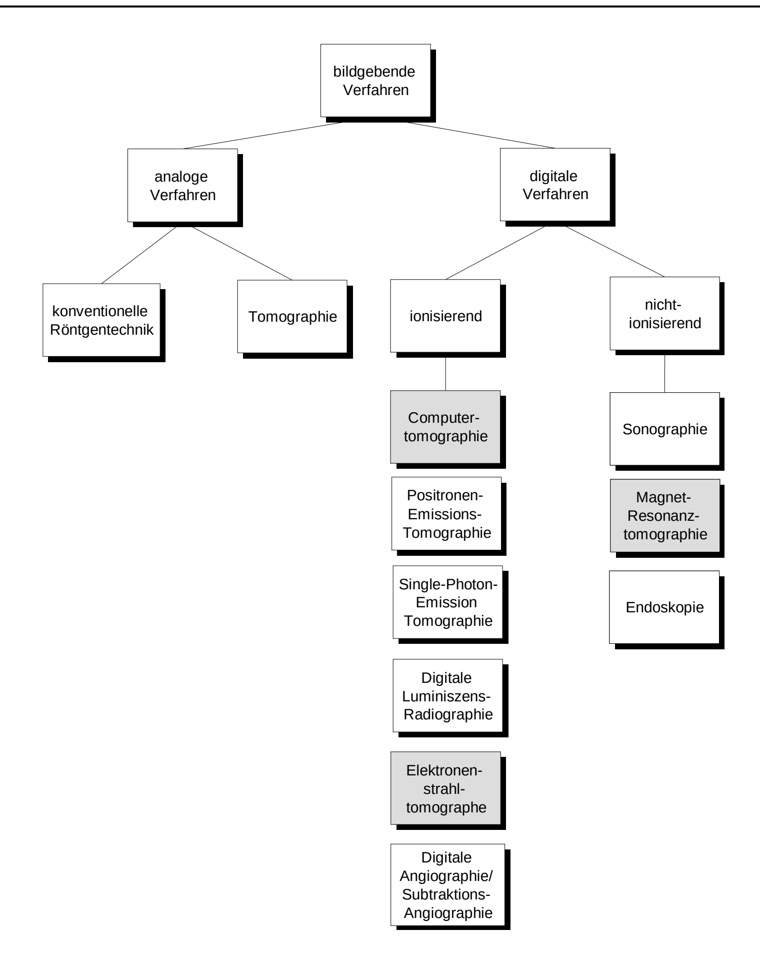


Abbildung 1 Bildgebende Verfahren in der Radiologie

Die bildgebenden Verfahren können nach der Form der Bildrepräsentation in analoge und digitale Verfahren eingeteilt werden.

Zu den analogen Verfahren zählt man unter anderem die konventionelle Röntgenbildtechnik. Dieses Verfahren stellt die Basis der radiologischen Diagnostik der Wirbelsäule dar und wird in der Praxis im Bedarfsfall durch digitale Verfahren ergänzt.

Der Vorteil der digitalen Verfahren liegt dabei in der direkten Weiterverarbeitung der radiologischen Bilder durch einen Rechner. Die digitale Form der Bilddaten ermöglicht z. B. die Bildverbesserung verrauschter Daten, diverse Visualisierungsmethoden (z. B. in Form von 3D-Darstellungen der Daten), die Bildanalyse (z. B. durch Segmentierung bestimmter Bildregionen) sowie die Übertragung bzw. Speicherung digitaler Bilddaten im Rahmen von Bildarchivierungs- und Kommunikationssystemen.

Bei den digitalen Untersuchungsverfahren unterscheidet man ionisierende (auf Röntgenstrahlung basierende) und nicht-ionisierende Methoden.

Unter den ionisierenden Methoden ist die Computertomographie die am häufigsten verwendete Ergänzung analoger Verfahren. Daneben bildet ein relativ neues Verfahren, die Elektronenstrahltomographie, die Möglichkeit, durch extrem kurze Scanzeiten Bewegungsartefakte, wie sie z. B. im Rahmen der Herzdiagnostik auftreten, weitgehend zu vermeiden.

Als einziger Vertreter aus der Gruppe der nicht-ionisierenden Verfahren wird die Magnetresonanztomographie genauer beschrieben.

3.1.1 Computertomographie

3.1.1.1 Verfahren

Im Unterschied zur konventionellen Röntgentechnik ist das erzeugte Bild kein Überlagerungsbild, sondern basiert auf einer lokalen physikalischen Eigenschaft des untersuchten Gewebes, dem Schwächungsvermögen der Röntgenstrahlung. Dieses läßt sich in Form von quantitativen Dichteangaben in Bezug zur Röntgenabsorption von Wasser (gemessen in Hounsfield-Einheiten) darstellen.

Im Grundaufbau besteht ein Computertomograph aus einer Röntgenröhre, einem Detektor, einem Steuerrechner und einem Bildrechner (siehe hierzu Abbildung 2).

Das Bildaufnahmesystem setzt sich aus der Röntgenröhre und einem Detektorfeld zusammen. Dabei sind beide Komponenten entweder gekoppelt und drehen sich gemeinsam um den Patienten oder die Detektoren befinden sich in einem geschlossenen feststehenden Ring um den Patienten, wobei die Röntgenröhre rotiert.

Während des Untersuchungsvorgangs wird im Rahmen der Meßwerterfassung für jede Winkelposition ein Absorptionsprofil erstellt. Aus der Gesamtheit der (eindimensionalen) Absorptionsprofile rekonstruiert ein Rechner nun mittels der sogenannten gefilterten Rückprojektion eine zweidimensionale Schichtaufnahme [Thurn et al. 1998].

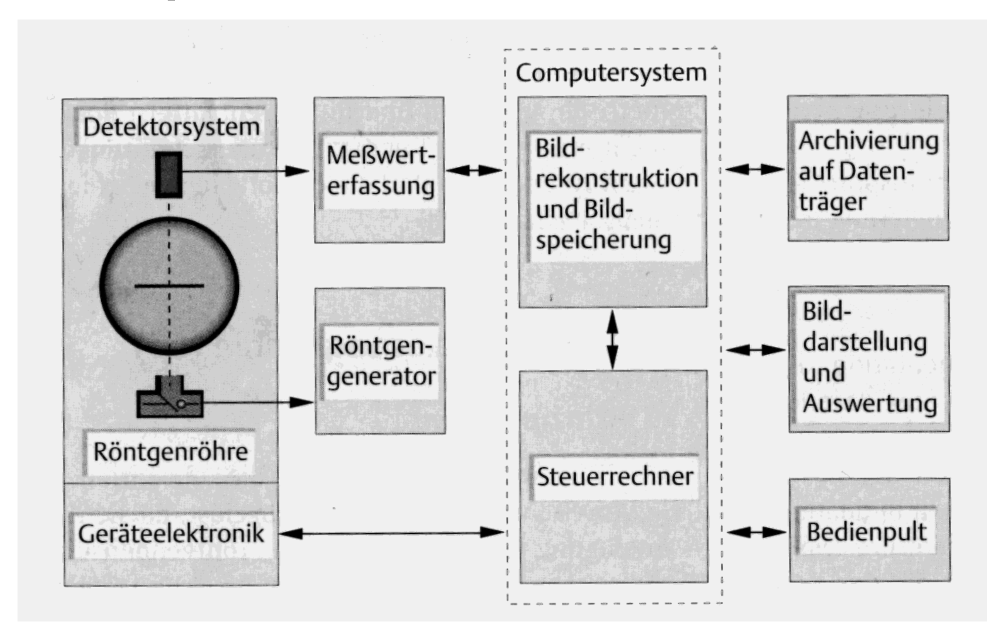


Abbildung 2 Aufbau eines Computertomographen

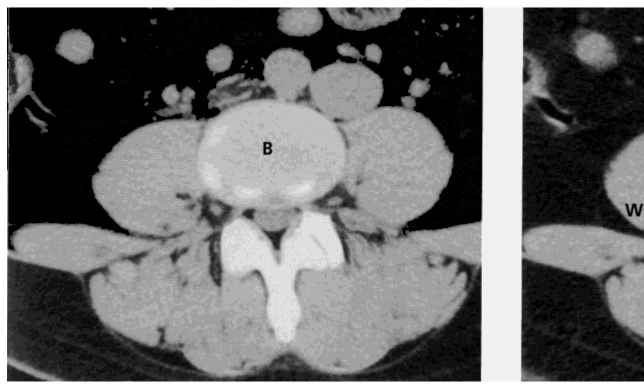
3.1.1.2 Anwendungsgebiet

Aufgrund der transversalen Darstellung und der verbesserten Beurteilbarkeit durch eine deutlich bessere Kontrastabstufung der dem Knochen benachbarten Weichteilstrukturen hat sich die Computertomographie in der Diagnostik von Skeletterkrankungen (insbesondere im Bereich der Wirbelsäule) etabliert und kann im Vergleich zu konventionellen Verfahren zusätzliche diagnostische Informationen bieten [Lissner et al. 1992].

Außerdem kann die Gewebsdichte nicht nur qualitativ (wie bei der konventionellen Röntgentechnik), sondern auch numerisch erfaßt und gespeichert werden. Dies ermöglicht z. B. neben der transversalen Darstellung der Ausgangsdaten die Rekonstruktion sagittaler oder koronarer Schichtaufnahmen ohne zusätzliche Belastung des Patienten.

Ein Nachteil des Verfahrens ist das im Vergleich zur konventionellen Tomographie deutlich geringere räumliche Auflösungsvermögen. Diese Eigenschaft macht sich vor allem bei der Darstellung von Knochenstrukturen negativ bemerkbar.

Indikationen für die Computertomographie sind u. a. Frakturen, Luxationen, Entzündungen und primäre bzw. sekundäre Knochentumoren [Burgener et al. 1997].



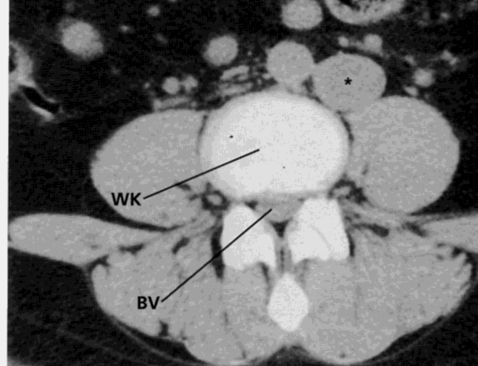


Abbildung 3 CT-Aufnahmen der Lendenwirbelsäule

3.1.2 Magnetresonanztomographie

3.1.2.1 Verfahren

Bei der Magnetresonanztomographie (synonym: Kernspintomographie) wird die Tatsache ausgenutzt, daß bestimmte Atome (u. a. Wasserstoff), bedingt durch sogenannte Kernspinbewegungen, ein magnetisches Gesamtmoment ungleich Null aufweisen. Dieser Effekt entsteht aufgrund eines Ladungsungleichgewichtes von Protonen und Neutronen. Die Ausrichtung der Atome kann durch Anlegen eines starken magnetischen Feldes beeinflußt werden. Dieses Feld führt zu einer gleichmäßigen Ausrichtung der ansonsten völlig ungeordneten Atomkerne. Nun wird ein Hochfrequenzimpuls bestimmter Stärke und Dauer in einem definierten Winkel zur z-

Achse (meist 90 oder 180°) der Atome eingestrahlt. Dieser Impuls ändert die Richtung der in der Probe vorhandenen magnetischen Momente mehr oder weniger stark. Ist der Impuls beendet, bewegen sich die Atomkerne unter Energieabgabe in ihre ursprüngliche Ausgangslage zurück. Während dieser als Relaxationszeit bezeichneten Phase geben die Kerne ein charakteristisches Hochfrequenzsignal ab, das registriert werden kann (siehe Abbildung 4). Die Relaxation der Atomkerne kann dabei in zwei Komponenten, nämlich die Spin-Gitter-Relaxationszeit (T1) und die Spin-Spin-Relaxationszeit (T2) aufgeschlüsselt werden.

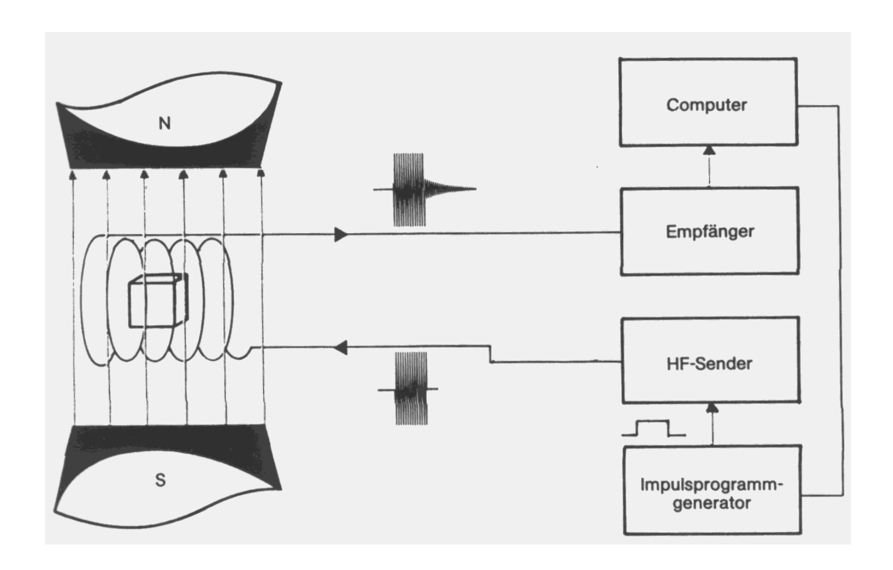


Abbildung 4 Erzeugung und Nachweis magnetischer Kernspinresonanz

Dadurch, daß das vom Atomkern ausgesandte Frequenzsignal proportional zur Protonendichte im angeregten Volumen ist und sich die Protonendichten einzelner Organe unterscheiden, kann aus diesen Informationen ein Bild erzeugt werden.

Da die Protonendichte jedoch nur geringe organspezifische Unterschiede aufweist werden sogenannte Gradientenspulen zur Kontrasterhöhung eingesetzt.

Einen Überblick über die wichtigsten Komponenten eines Magnetresonanztomographen (insbesondere die Magnet- und Gradientenspulen, ein System zum Senden und Empfangen von Hochfrequenz-Signalen sowie ein zentraler Steuerrechner) gibt Abbildung 5.

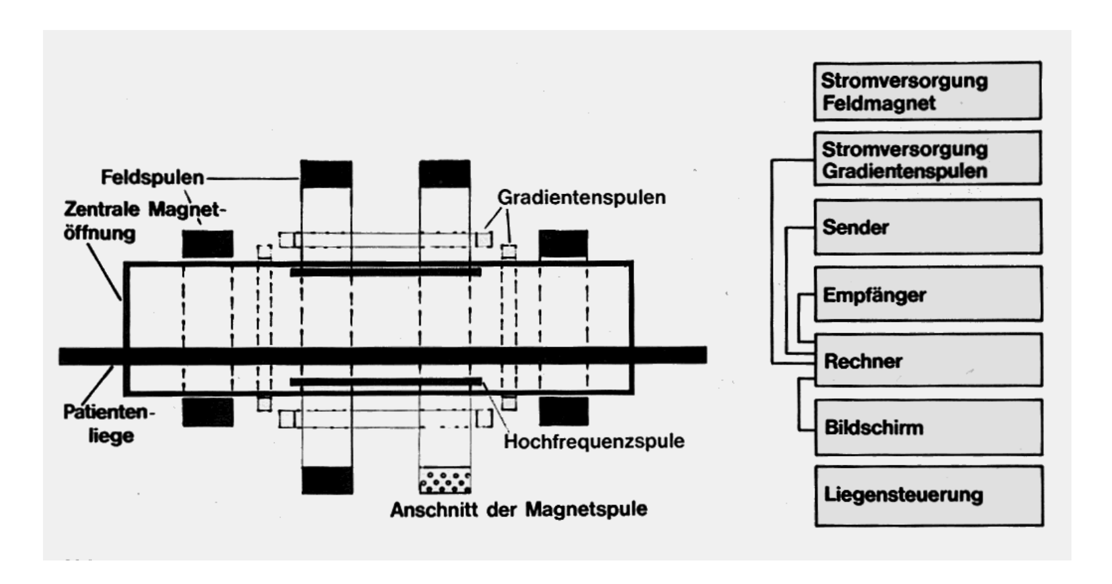


Abbildung 5 Aufbau eines Magnetresonanztomographen

3.1.2.2 Anwendungsgebiet

Der Vorteil der Kernspintomographie ist neben der multiplanaren Schichtführung (durch eine beliebige Orientierung des statischen Magnetfeldes) und einer hohen Kontrastauflösung der für den Patienten weitgehend belastungsfreie Untersuchungsvorgang.

Indiziert ist das Verfahren u. a. bei Gelenk- und Bandscheibenläsionen, Knochennekrosen, Osteomyelitis und anderen entzündlichen Prozessen sowie primären und sekundären Knochentumoren bzw. dem Knochen benachbarten Weichteiltumoren [Lissner et al. 1992].

Dabei liegt die Stärke dieses Verfahrens weniger in der Darstellung ossärer Feinstrukturen (aufgrund der geringen Protonendichte der Knochenstrukturen), sondern eher in der Abbildung von Weichteilstrukturen der gesamten Wirbelsäule [Thurn et al. 1998].

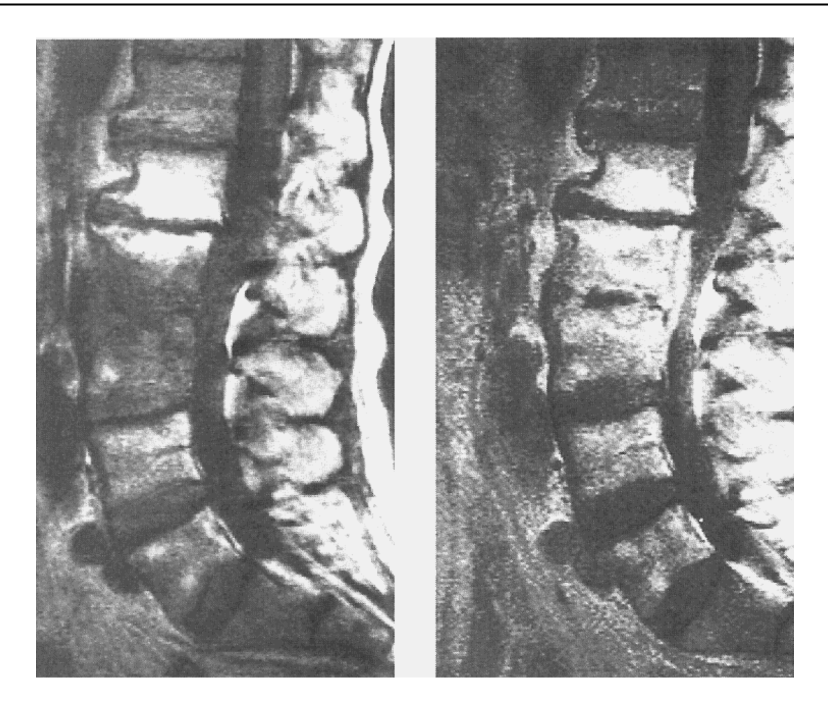


Abbildung 6 MRT der Lendenwirbelsäule

3.1.3 Elektronenstrahltomographie

3.1.3.1 Verfahren

Im Gegensatz zur konventionellen Computertomographie besitzt die Elektronenstrahltomographie (Electron Beam Tomography, kurz: EBT) keine rotierende Röntgenröhre. Der Elektronenstrahl wird vielmehr über Fokussierungs- und Ablenkspulen auf die halbkreisförmig um den Patienten angeordneten Wolframtargets gelenkt. Diese erzeugen die eigentliche Röntgenstrahlung „vor Ort“, die wiederum von entsprechenden Detektorringen empfangen wird (siehe Abbildung 7). Da die Ablenkung des Elektronenstrahls elektromagnetisch erfolgt, sind extrem kurze Scanzeiten (Abtastzeiten) im Millisekunden-Bereich realisierbar. Insgesamt entspricht das Prinzip des EBT dem einer riesigen konventionellen Röntgenröhre.

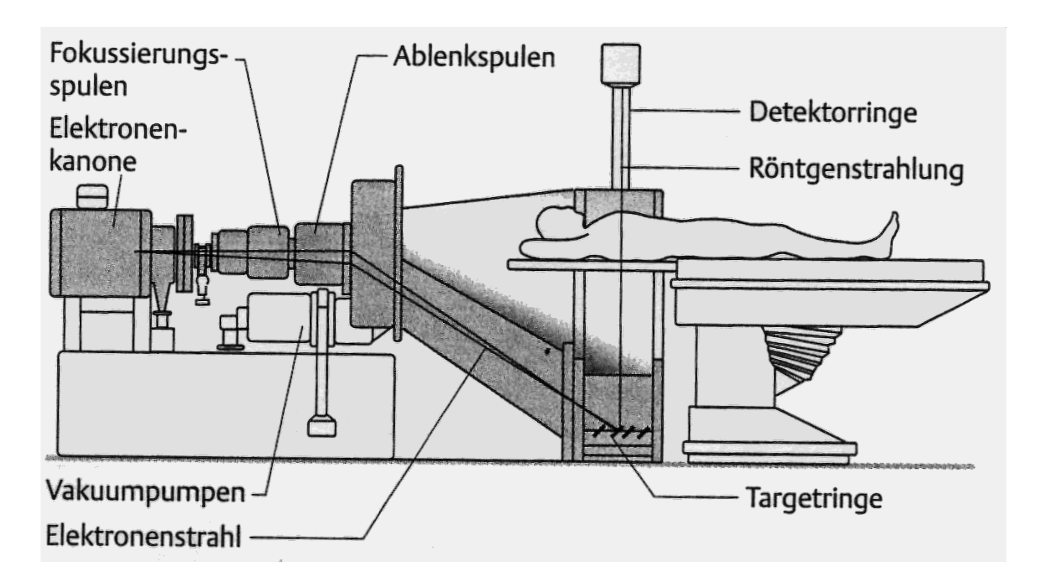


Abbildung 7 Prinzip der Elektronenstrahltomographie

3.1.3.2 Anwendungsgebiet

Durch die verkürzte Scanzeit findet das Verfahren unter anderem Anwendung in der Untersuchung von nicht-kooperativen Patienten (Pädiatrie, Notfalldiagnostik).

Des weiteren wird das Verfahren bei der Studie an bewegten Organen angewendet. Für derartige Untersuchungen ist die konventionelle Computertomographie ungeeignet, da in diesen Fällen häufig Bewegungsartefakte auftreten.

Der Hauptanwendungsbereich der Elektronenstrahltomographie ist jedoch die Diagnostik der Funktion und der Morphologie des Herzens und der Koronargefäße. Durch die geringen Scanzeiten wirken sich die Pulsationen des Herzens während der Untersuchung nicht störend aus.

Neben pathologischen Veränderungen (z. B. Kalzifikationen) können mittels Computerauswertung auch quantitative Funktionsparameter wie z. B. Perfusion, Kammervolumen etc. ermittelt werden.

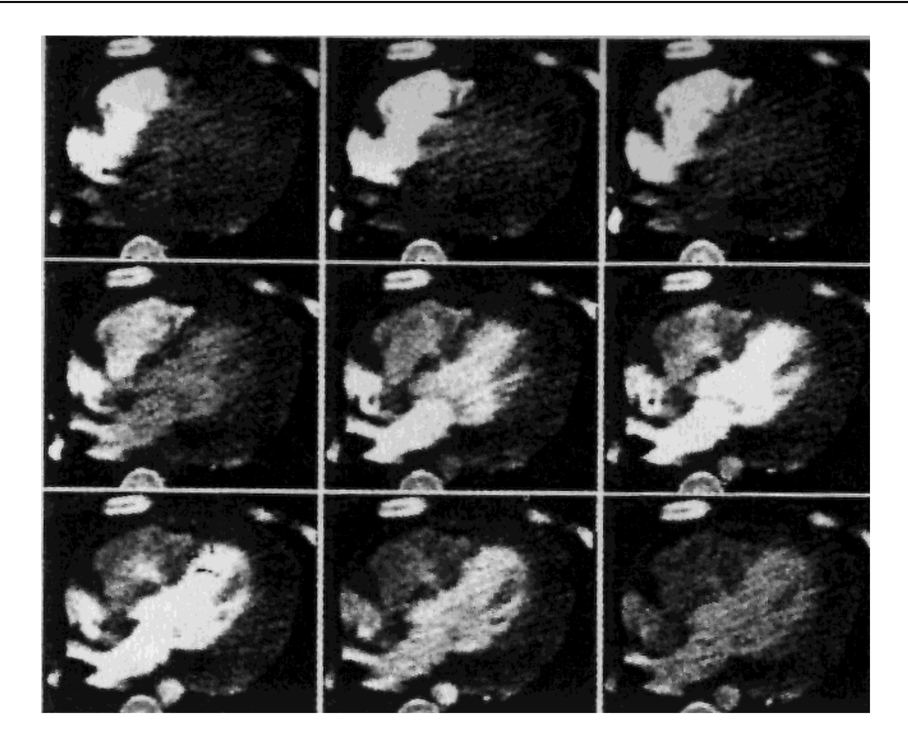


Abbildung 8 Darstellung der Herzphasen mittels EBT

3.2 Die radiologische Befundung

3.2.1 Die radiologische Fachsprache

Sprache ist allgemein definiert als „ein System von gesprochenen bzw. geschriebenen Zeichen zum Zwecke der Kommunikation, die nach festgelegten Gesetzmäßigkeiten zu größeren Einheiten wie etwa Silben, Wörtern, Phrasen, Sätzen oder ganzen Texten kombiniert werden“[Seelos et al. 1997].

Zur Einordnung der radiologischen Fachsprache ist eine generelle Aufteilung der Sprache in natürliche und formale Sprachen hilfreich.

Natürliche Sprachen sind historisch gewachsene Sprachen, deren Regeln den Gebrauch widerspiegeln, ohne notwendigerweise bekannt zu sein.

Formale Sprachen (z. B. Programmiersprachen) sind hingegen konstruierte Systeme, die aus einer festgelegten Menge grammatikalischer Regeln ableitbar sind.

Die radiologische Fachsprache kann zwischen den natürlichen und den formalen Sprachen eingeordnet werden. Einerseits teilt sie wichtige Struktureigenschaften mit einer Reihe natürlicher Sprachen, wie z. B. der deutschen, der lateinischen oder der griechischen Sprache. Andererseits zeichnet sie sich durch eine, für Fachsprachen symptomatische hohe Kompaktheit und Präzision aus. Diese Eigenschaft teilt sie wiederum mit den formalen Sprachen.

Nachfolgend sollen die unterschiedlichen linguistischen Beschreibungsebenen einer Sprache erläutert werden. In Hinblick auf den im System verwendeten Ansatz (Lernen

der Syntax aus einer vorgegebenen Menge von Befundungstexte) beschränken sich die folgenden Betrachtungen schwerpunktmäßig auf die Syntax einer Sprache.[1]

Die Lehre von der Struktur der Sätze und ihrer Teile wird als *Syntax* bezeichnet. Sie beschäftigt sich mit den Regeln zum Aufbau größerer sprachlicher Einheiten (etwa Sätzen) aus kleineren Einheiten (etwa Worten). Einzelne Worte besitzen dabei bestimmte morphologische Merkmale, wie z. B. Wortart, Kasus, Numerus und Genus. Eine Grammatik beschreibt dann eine Menge von Regeln zur sinnvollen und korrekten Kombination von Worten zu größeren Einheiten (z. B. Sätzen). Eine grammatikalische Analyse eines sprachlichen Ausdrucks auf syntaktische Korrektheit wird dabei als Parsing bezeichnet.

Ein grammatikalisch falscher Ausdruck wäre z. B. „*Chronisches Status vor Microtherapie*“, wohingegen „*Chronischer Status vor Microtherapie*“ grammatikalisch korrekt wäre.

Die *Semantik* untersucht die Bedeutung von größeren sprachlichen Einheiten und orientiert sich dabei an der syntaktischen Struktur. Des weiteren stellt sie sicher, daß nur sinnvolle, syntaktisch korrekte Ausdrücke erzeugt werden können.

Die *Pragmatik* beschränkt sich im Gegensatz zu den beiden vorherigen Beschreibungsebenen einer Sprache nicht ausschließlich auf sprachinterne Phänomene, die allein am Textmaterial ausgerichtet sind, sondern betrachtet das Umfeld bzw. den Kontext in dem eine sprachliche Äußerung erfolgt. Die Pragmatik bezüglich der radiologischen Befundung ist dabei eindeutig festgelegt, und der sprachliche Kontext ist in diesem Fall die objektive Beschreibung eines radiologischen Bildes.

Zur detaillierteren Charakterisierung radiologischer Fachsprachen sollen im folgenden deren wichtigste Eigenschaften beschrieben werden. (siehe [PG 281 1996])

Ein stilistisches Merkmal radiologischer Fachsprachen ist der sogenannte Telegrammstil. Er zeichnet sich dadurch aus, daß radiologische Befundungstexte selten vollständige Sätze, sondern eher Standardfloskeln („kein Nachweis...“) bzw. verblose Satzfragmente („Regelrechter Abschluß der Bandscheiben in der Dorsalkante des Wirbelkörpers“) enthalten.

Zum anderen ist häufig eine Verdichtung (Kondensierung) der radiologischen Fachsprache zu verzeichnen, die sich unter anderem in der Bildung von Komposita manifestiert und zu Wortneubildungen wie z. B. „Facettengelenkarthrose“ führt.

Außerdem ist ein hoher Grad an Redundanz zu verzeichnen. Beispiele hierfür finden sich im vorliegenden Befundungstext z. B. an folgenden Stellen: „Bandscheibenvorfall“ (Abschnitt C7/Th1) und „Bandscheibenprolaps“ (Abschnitt C5/6), bzw. „Die

---

[1] Für eine intensivere Beschäftigung mit den grundlegenden Begriffen und Prinzipien der automatischen Sprachverarbeitung sei an dieser Stelle auf [Hellbig 1991] verwiesen.

Neuroforamina und der Spinalkanal sind *unauffällig*“ (Abschnitt C3/4) und “Die Neuroforamina und der Spinalkanal sind *nicht eingeengt*“ (Abschnitt C4/5). Das Problem der Redundanz kann weitgehend durch eine Standardisierung radiologischer Begriffe umgangen werden. Eine Umsetzung dieser Standardisierung wird im vorliegenden System im übrigen durch Verwendung eines Wörterbuches realisiert.

Medizinische Fachsprache im Allgemeinen und die radiologische Fachsprache im Speziellen erlaubt die Beschränkung auf bestimmte sprachliche Phänomene. Diese Eigenschaft der Fachsprachen hat zur Folge, daß Sprache in ihrer vollen Allgemeinheit nicht betrachtet werden muß [Schröder 1993]. Formal ist die der radiologischen Fachsprache zugrundeliegende Grammatik weniger umfangreich als die eines beliebigen Textes. Diese Eigenschaft wirkt sich entsprechend positiv auf das Ergebnis des Lernverfahrens aus.

Des weiteren wird durch die Beschränkung eines konkreten Befundungsvorganges auf eine bestimmte Domäne jeweils nur ein eingeschränkter Ausschnitt der radiologischen Fachsprache verwendet.

Die dargestellten Eigenschaften der radiologischen Fachsprache sowie die nachfolgend beschriebene Struktur radiologischer Befundungstexte ermöglichen, mittels der verwendeten maschinellen Lernverfahren (siehe Kapitel 5) aus Befundungstexten einer bestimmten Domäne ein geeignetes System zur Befundungsunterstützung zu generieren, das dem Arzt eine strukturierte Eingabe neuer Befunde ermöglicht.

3.2.2 Der radiologische Befund

[PG 281 1996] beschreibt die Intention radiologischer Befundungstexte als „... die möglichst explizite und eindeutige Beschreibung eines radiologischen Bildes. Im Idealfall ist ein solcher Befund vollkommen objektiv. ... Befundungstexte beschreiben Normalzustände ebenso wie pathologische Zustände.“

Die obige Definition soll den Zusammenhang zwischen dem radiologischen Bild, der im Befund verwendeten radiologischen Sprache und der Struktur des Befundungstextes verdeutlichen.

3.2.2.1 Der Aufbau eines radiologischen Befundes

Klinische Dokumentationsobjekte wie z. B. Aufnahmeformulare, Anamnesebögen, Arztbriefe, Laboranforderungen, Konsiliarberichte, Operationsberichte, Befundberichte, etc. weisen teilweise erhebliche Variationen in Art und Umfang der Strukturierung auf.

So unterscheidet [Leiner et al. 1998] Dokumente mit schwachen strukturellen Vorgaben (z. B. Arztbriefe, Operationsberichte) von Dokumenten mit starken strukturellen Vorgaben (z. B. Aufnahmeformulare, Befundberichte etc.).

Im Idealfall eines strukturierten Befundes (siehe Abbildung 9) kann man auf der obersten Abstraktionssebene eine dreiteilige Gliederung in einen Briefkopf, den eigentlichen Befundtext und einen Brieffuß erkennen.

Der Briefkopf setzt sich aus dem Adressaten des Befundes, dem aktuellen Datum und einer Begrüßungsfloskel zusammen.

Danach folgt je nach Domäne (z. B. Computertomographie, Magnetresonanztomographie oder Elektronenstrahltomographie) eine vorgegebene Menge von Befundangaben (im folgenden als Abschnitte bezeichnet). Diese Abschnitte werden in der Regel durch ein gesondertes Trennzeichen (hier Doppelpunkt) bzw. durch spezielle Formatierungen (hier Fettdruck) hervorgehoben.[2]

Die vorgegebene Strukturierung der Befunde durch Aufteilung in einzelne Abschnitte wird, wie bereits erwähnt, beim Aufbau des entsprechenden Software-Agenten verwendet (Eine ausführlichere Diskussion dieses Strukturierungs-Konzeptes findet sich in Kapitel 8.3.1.)

Der konkrete Befund beginnt mit der Angabe des Untersuchungsdatums sowie der Art und Lokalisation der durchgeführten Untersuchung (Abschnitt *Betreff*). Danach wird die Fragestellung der Untersuchung beschrieben. Es folgt eine präzise Darstellung der angewandten Untersuchungstechnik (z. B. im konkreten Fall der Computertomographie die Angabe der Schichtdicke, der Lokalisation der Untersuchung und der verwendeten Fenstertechnik). Anschließend folgen die eigentlichen Befunddaten.

Im konkreten Fall werden die einzelnen Segmente der Halswirbelsäule des Patienten (hier von C2/3 bis C7/Th1) im Detail beschreiben. Dabei werden sowohl pathologische Auffälligkeiten („...Arthrose der Facettengelenke.“) als auch unauffällige Befunde innerhalb eines Segmentes (z. B. „...Kein Nachweis eines Bandscheibenprolaps.“) bzw. komplett unauffällige Segmente (Segment C2/3) festgehalten (siehe hierzu auch die Definition in Kapitel 3.2).

Der Aufbau des beschriebenen Befundtextes soll nur als Rahmen für eine bestimmte Klasse radiologischer Befunde (in diesem Fall die Computertomographie der Halswirbelsäule) verstanden werden. So weisen unterschiedliche Klassen bzw. unterschiedliche Domänen in der Regel eine mehr oder weniger große Varianz in der Auswahl respektive der Abfolge einzelner Abschnitte auf. So sind in der Befundung von Elektronenstrahltomographien des Herzens andere Befundangaben von Bedeutung (koronare Risikofaktoren, Kalkläsionen je Segment etc.) als z. B. bei der Magnetresonanztomographie der Wirbelsäule. Aus diesem Grund erreicht man mit dem vorgestellten System die besten Ergebnisse, wenn man für jede Domäne einen speziellen Automaten lernt.

---

[2] Diese Eigenschaft der Befundtexte eröffnet im übrigen die Möglichkeit zur automatischen Übertragung dieser Strukturierung auf die vom System erzeugte Datenstruktur. Zum besseren Verständnis dieses Vorgangs sei an dieser Stelle auf das Kapitel 7 Verarbeitung von Befundtexten verwiesen.

Institut für MikroTherapie, Universitätsstr. 142, D - 44799 Bochum

Herrn Mustermann
Musterstr. 10

12345 Musterstadt

Bochum, 20.12.1998, mm

Sehr geehrter Herr Mustermann,

Betreff: 17.12.1998: Computertomographie Halswirbelsäule

Fragestellung:

Status vor Mikrotherapie.

Untersuchungstechnik:

Kontinuierliche, abschlußplattenparallele 3/3-mm-Schichtung der Intervertebralräume C3/4 bis C7/Th1, Ausdruck in Weichteil- und Knochenfenstertechnik.

Segment C2/3:

Unauffällig.

Segment C3/4:

Regelrechter Abschluß der Bandscheiben in der Dorsalkante des Wirbelkörpers. Die Neuroforamina und der Spinalkanal sind unauffällig.

Segment C4/5:

Regelrechter Abschluß der Bandscheiben in der Dorsalkante des Wirbelkörpers. Die Neuroforamina und der Spinalkanal sind nicht eingeengt.

Segment C5/6:

Medialer bis links lateraler Bandscheibenprolaps mit Pelottierung und leichter Kompression des Duralschlauchs. Einengung des linken Neuroforamens. Spondylosis deformans.

Segment C6/7:

Dorsale Bandscheibenprotrusion mit Pelottierung des Duralsacks. Spondylosis deformans mit ventraler und dorsaler Osteophytenbildung. Knöcherne Einengung des rechten Neuroforamens. Facettengelenkarthrose.

Segment C7/Th1:

Kein Nachweis eines Bandscheibenvorfalls. Arthrose der Facettengelenke. Spondylosis deformans.

Mit freundlichen Grüßen

Dr. Muster

Abbildung 9 Beispiel eines radiologischen Befundes

3.2.3 Kontext der radiologischen Befundung

3.2.3.1 Die universitäre Ausbildung

Die radiologische Ausbildung im Rahmen des Medizinstudiums beschränkt sich auf zwei einsemestrige Kurse in Strahlenschutz sowie in radiologischer Diagnostik. Begleitet werden diese Kurse dabei jeweils von einer einstündigen Vorlesung. Diese Kurse bieten jedoch in der Regel nicht die Möglichkeit, die erworbenen theoretischen Grundlagen in der Praxis anzuwenden und das eigenständige Befunden in ausreichendem Maße zu trainieren.

Das im Rahmen der Diplomarbeit erstellte System bietet hingegen die Möglichkeit aus bereits erstellten Befunden, die einer bestimmten Domäne bzw. einer bestimmten Untersuchungstechnik zuzuordnen sind, eine Benutzeroberfläche zu generieren, die die Struktur der Befunddaten wiedergibt. Diese Benutzeroberfläche gewährleistet, bei entsprechenden Eingabedaten, daß der Student strukturiert durch den Befundungsvorgang geleitet wird.

3.2.3.2 Praxis der radiologischen Befundung

Im Rahmen der Befunderfassung wird zwischen konventionellen (nicht rechner-basierten) und rechnerbasierten Methoden unterschieden.

Bei der konventionellen Befunderfassung werden die Befunddaten zum Teil handschriftlich oder mittels Diktat erfaßt und müssen zur Weiterverarbeitung im Rechner nachträglich von einer Schreibkraft in eine elektronische Form übertragen werden. Dieser Vorgang dauert zum einen wesentlich länger als die direkte Eingabe des Befundes in den Rechner, zum anderen ist die Gefahr einer Verfälschung der ursprünglichen Daten (aufgrund des mehrfachen Medienbruches) relativ hoch.

Bei der rechnerbasierten Befunderfassung werden die Befunddaten direkt in den Rechner eingegeben. Die Eingabe der Befunde kann dabei durch ein Befundungs-unterstützungs-System (siehe Kapitel 3.3) erleichtert werden.

In der Praxis ist der Vorgang der Befundung in vielen Fällen noch nicht standardisiert und bietet Raum für Verbesserungen hinsichtlich der Qualität und der Verfügbarkeit der erstellten Befunde.

Das vorgestellte System setzt an diesen Punkten an, um die Effizienz des Befundungsvorgangs zu erhöhen.

3.3 Befundungsunterstützungs-Systeme

3.3.1 Generelle Vorteile von Befundungsunterstützungs-Systemen

Bei der Befunderstellung mittels Freitext oder Diktat muß der Befund erst in eine elektronische Form übertragen werden. Dieser Prozeß bedeutet einen hohen Zeitaufwand und die Gefahr der Verfälschung der ursprünglichen Daten.

Ein wichtiger Aspekt beim Einsatz von Befundungsunterstützungs-Systemen ist die deutliche Verbesserung der Vollständigkeit der erstellten Befunde. Diverse Studien (z. B. [Heyder et al. 1988] sowie [Kuhn 1994]) haben gezeigt, daß der Anteil fehlender Angaben in einem freitextlich erstellten Befund oft einen hohen Prozentsatz ausmacht und daß dieser Anteil unter der Vorgabe einer strukturierten Erfassung der Befundtexte signifikant abnimmt.

Des weiteren erhöht sich auch die Objektivität, d. h. die Vereinheitlichung der erstellten Befunde. Befunde, die mittels nicht-standardisierter Befundungsverfahren erstellt wurden weisen oft eine hohe Variationsbreite bei der Beschreibung der Befunddaten auf und es besteht kaum Konsens darüber, in welchem Detaillierunggrad bzw. mit welcher Terminologie dokumentiert werden soll. [Kuhn 1996]

Ein weiterer Gesichtspunkt der radiologischen Befundung ist die Validität der Befunde, also ihre Richtigkeit, Zuverlässigkeit und Konsistenz im Kontext der medizinischen Qualitätssicherung. Dieser Aspekt ist unter der Voraussetzung, daß die zur Erstellung des Systems verwendeten Befunddaten die genannten Eigenschaften besitzen erfüllt.

Außerdem erleichtert die direkte elektronische Verfügbarkeit der Befunddaten z. B. die Integration in ein radiologisches Informationssystem. (siehe hierzu [Lipinski 1999])

Hinzu kommt ein Trainingseffekt für weniger erfahrene Ärzte bzw. Studenten, da diese gezielt und strukturiert durch den Befundungsvorgang geleitet werden.

Als eines der wichtigsten Argumente gegen eine strukturierte, computerunterstützte Dokumentation wird häufig die Einschränkung bei der Befundformulierung angesehen, die eine Umstellung der Befundungsgewohnheiten des befundenden Arztes und dadurch einen möglicherweise erhöhten Zeitbedarf nach sich ziehen. Diesem Argument wurde jedoch bei der Konzeption des entwickelten Systems in Form einer umfassenden Konfigurierbarkeit und Adaptivität Rechnung getragen.

3.3.2 Textbaustein-basierte Systeme

Bei Textbaustein-basierten Systemen[3] werden die Befundungstexte aus vorgefertigten Sätzen zusammengefügt. Diese Sätze müssen mit entsprechendem Aufwand aus konventionell erstellten Befunden extrahiert und entsprechend aufbereitet werden (z. B. durch Zusammenfassung von Synonymen, Gruppierung von Satzfragmenten, etc.).

---

[3] Ein Beispiel für ein Texbaustein-basiertes Befundungsunterstützungs-System ist z. B. [Beran 1984]

Da diese Sätze bzw. Satzfragmente aber in der Regel keinen direkten Bezug zueinander haben, muß eine gewünschte Strukturierung explizit vorgegeben werden, da der Benutzer ansonsten nicht strukturiert durch den Befundungsvorgang geleitet werden kann. In diesem Zusammenhang besteht auch die Gefahr, daß vor allem ungeübte Benutzer unstrukturierte, unter Umständen sogar unsinnige Eingaben vornehmen. Ebenso ist eine Änderung der vorgefertigten Sätze teilweise schwierig bzw. kann erst im schon erstellten Befund vorgenommen werden. Außerdem müssen für jede neue radiologische Domäne explizit entsprechende Textbausteine erstellt bzw. bestehende erweitert werden.

3.3.3 Wissensbasierte Systeme

Wissensbasierte Systeme bestehen aus einer Reihe von interagierenden Komponenten. Die einzelnen Komponenten sind das Wissenserwerbsmodul, das Problemlösungsmodul, die Erklärungskomponente, die Wissensbasis und die Dialogkomponente.

Die Dialogkomponente steuert die Interaktion mit dem Benutzer, insbesondere die Eingabe von Daten bezüglich des aktuell zu lösenden Problems, sowie die Ausgabe der vom System ermittelten Problemlösung.

Das Problemlösungsmodul versucht, anhand der eingegebenen Fakten und des vorhandenen Wissens aus der Wissensbasis das jeweilige Problem des Anwenders zu lösen. Die Problemlösungskomponente wird auch als Inferenzmaschine, der Problemlösungsprozeß als Inferenz bezeichnet.

Die Erklärungskomponente vermittelt das für eine Problemlösung angewandte Vorgehen und macht die Schlußweise des Systems für den Anwender transparent.

Das Wissenserwerbsmodul ermöglicht das Auffüllen und die Abfrage der Wissensbasis und unterstützt die Akquisition des Expertenwissens.

Die Qualität eines wissensbasierten Systems hängt nun maßgeblich von der Wissensbasis ab. Der Vorgang der Wissensakquisition wird somit zu einem der entscheidenden Schritte bei der Konzeption und dem Aufbau eines solchen Systems.

Die Erfahrung in der Erstellung wissensbasierter Systeme hat gezeigt, daß gerade der Vorgang der Wissensakquisition oft den „Flaschenhals“ im Entwicklungsprozeß darstellt.

[Altenkrüger et al. 1992] beschreiben dieses Problem wie folgt: „...Die Erfahrung aus vielen Projekten zeigt, daß das Wissen von Experten zur Lösung einer Aufgabe häufig nicht bewußt vorhanden ist, sondern unbewußt angewandt wird. Die Ermittlung und Strukturierung des Wissens gestaltet sich häufig deutlich schwieriger als zuvor angenommen. Aber gerade das Resultat der Wissensakquisition ist für den Erfolg oder Mißerfolg wissensbasierter Systeme entscheidend.“

Die Erstellung einer konsistenten und vollständigen Wissensbasis stellt somit eine große Herausforderung bei der Entwicklung wissensbasierter Systeme dar und ist in der Regel nur durch enge Kooperation von mehreren Fachexperten möglich [Boegl et al. 1995].

Der Nachteil wissensbasierter Systeme im praktischen Einsatz[4] liegt vor allem darin, daß in der Regel für jede einzelne (radiologische) Domäne die Modellierung des Wissens explizit durchgeführt werden muß und nicht wie bei dem vorgestellten System ein domänenunabhängiges Verfahren verwendet wird, das eine entsprechend aufwendige Wissensakquisition überflüssig macht.

[Bernauer 1991] entwickelte ein System zur Unterstützung des Befundungsvorganges, bei dem die Befundaussagen mit Hilfe einer graphischen Benutzeroberfläche zusammengestellt und anschließend als Text ausgegeben werden können. Dabei beschränkt ein Domänenmodell die Eingabemöglichkeiten, die über die Benutzeroberfläche vorgenommen werden können, wobei die Semantik einer Äußerung hierbei mittels eines Conceptual Graphs realisiert wird. Auch hier stellt sich wieder das Problem, das Wissen, das in Form eines Conceptual Graphs repräsentiert wird, entsprechend zu akquirieren.

[4] Eine ausführlichere Betrachtung wissensbasierter Systeme in der Medizin findet sich z. B. in [Adlassnig 1993] und [Berner et al. 1999]

4 Intelligente Softwareagenten

4.1 Begriff des Softwareagenten

Dieses Kapitel soll als Einführung in den Begriff der intelligenten Softwareagenten dienen. Gegenwärtig existieren hierfür zahlreiche unscharfe und teilweise sich überschneidende Begriffsdefinitionen.

Eine wesentliche Eigenschaft eines Softwareagenten ist seine Intelligenz. Ohne auf die weitere Definition von Intelligenz eingehen zu wollen, läßt sich feststellen, daß die Fähigkeit des Lernens im Allgemeinen als intelligent angesehen wird.

Der in der deutschsprachigen Literatur verwendete Begriff des Agenten beruht auf der inkorrekten Übersetzung des englischen Begriffes "agent" [Lüth, 1998], was eigentlich soviel wie "Handelnder" bzw. "wirkende Kraft" [Langenscheidt, 1991] bedeutet und von dem lateinischen Verb "agere" abgeleitet ist [Stowasser, 1980], welches sich im Deutschen noch in den Begriffen "agieren" und "Agentur" findet.

Ein intelligenter Agent ist laut Brenner eine "Software, die in einer digitalen, vernetzten Umgebung selbständig Aufgaben im Auftrag des Benutzers erledigt" [Brenner et al., 1998].

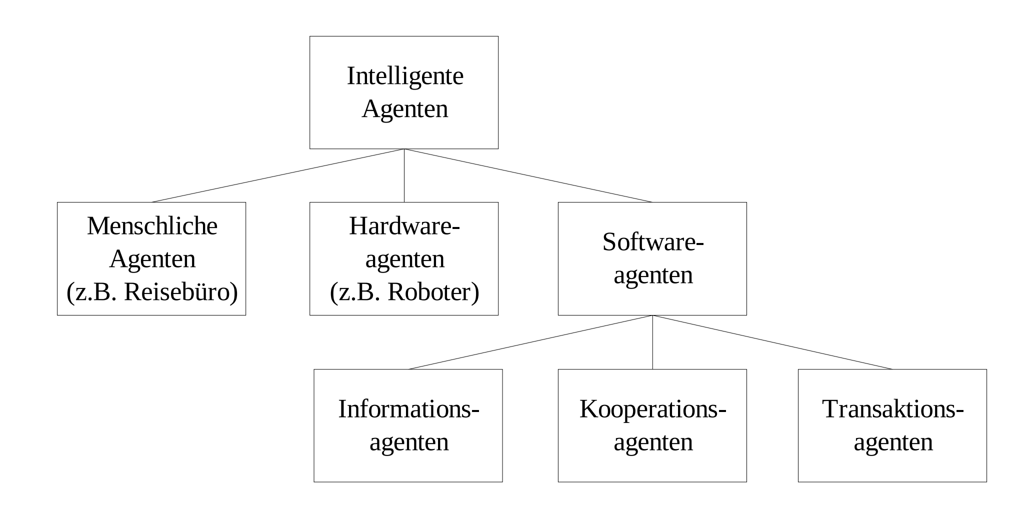


Abbildung 10 Kategorien intelligenter Agenten

Zu einer ersten Einteilung des breiten Spektrums von Intelligenten Agenten lassen sich drei Kategorien finden, die von unterschiedlicher Beschaffenheit sind: Menschliche Agenten, Hardwareagenten und Softwareagenten.

Ein menschlicher Agent ist im Prinzip jede Person, die einer Dienstleistungstätigkeit nachgeht. Als Beispiel sei hier ein Angestellter eines Reisebüros zu nennen. Der Kunde wird bezüglich möglicher Reiseziele beraten oder nennt direkt sein Wunschziel. Daraufhin erhält er ein Angebot über eine Unterkunft, Verpflegung und die Reiseroute. Sämtliche mit dem Buchen einer Reise verbundenen Aufgaben werden von dem "travel agent" ausgeführt.

Dieses Bild läßt sich auch auf die hier näher betrachteten Softwareagenten übertragen. Der Nutzer nennt seine Wünsche, und der Agent hilft ihm bei deren Erfüllung, oder sie werden direkt von ihm erfüllt.

Softwareagenten weisen Ausprägungen als Informations-, Kooperations- und Transaktionsagenten auf, die im folgenden erläutert werden:

Informationsagenten unterstützen den Benutzer bei der Beschaffung von Informationen, die für ihn relevant sind. Beispiele dafür sind Internet-Suchmaschinen, die z.B. personalisierte Nachrichten liefern (PointCast Network) oder den jeweils günstigsten Anbieter von Musik-CDs herausfinden (BargainFinder) (siehe [Caglayan et al. 1997]).

Kooperationsagenten hingegen versuchen, eine komplexe Aufgabe durch Kommunikation und Kooperation mit Agenten, Menschen oder anderen Ressourcen zu lösen. Kooperationsagenten werden benutzt, wenn die Lösung für einen Agenten zu komplex wäre, oder wenn das Wissen von bereits existierenden Agenten genutzt werden kann, um zu einer Lösung zu gelangen.

Transaktionsagenten schließlich führen Transaktionen aus oder überwachen diese, wie z.B. Kauf von Ware beim elektronischen Handel.

Die oben beschriebenen Ausprägungen der Softwareagenten sind keineswegs disjunkt zu sehen. Es kann Agenten geben, die sogar alle drei vorweisen können. Zum Beispiel wäre es denkbar, einen Agenten mit dem Kauf einer CD eines Künstlers zu beauftragen, deren Titel man nicht kennt.

Der Agent hätte jetzt die Aufgabe, zuerst Informationen über den Künstler und seine Werke zu finden. Hierbei könnte er auf Informationsagenten zugreifen und anschließend andere nutzen, um den günstigsten Anbieter herauszufinden. Ist dies geschehen, so kann dort die CD bestellt werden. Evtl. wäre es dem Agenten auch möglich, die gewünschte CD bei einer Internetauktion oder einem virtuellen Marktplatz von anderen Softwareagenten oder realen Bietern zu erstehen.

4.2 Charakteristika

Um von einem intelligenten Softwareagenten reden zu können, müssen einige Charakteristika erfüllt sein, die ihn gegen herkömmliche Softwareprodukte abgrenzen.

Zu unterscheiden sind hier interne und externe Charakteristika. Auch hier sind die Grenzen nicht eindeutig. Interne Charakteristika betreffen nur den Agenten und seinen Aufbau; Externe beschreiben, wie der Agent mit seiner Umwelt umgeht. Als Umwelt für Agenten können andere Agenten, Interaktion mit dem Benutzer oder andere Informationsquellen dienen.

4.2.1 Reaktivität

Der Agent reagiert angemessen auf seine Umwelt. Hierfür muß er eine geeignete Repräsentation und Sensoren aufweisen.

4.2.2 Proaktivität/Zielorientiertheit

Der Agent reagiert nicht nur auf Umweltveränderungen, sondern agiert auch selbständig. Hierzu benötigt er i.A. ein komplexes Ziel, für dessen Erfüllung er aktiv wird.

4.2.3 Schlußfolgerunngs-/Lernfähigkeit

Die Intelligenzleistung eines Agenten beruht im wesentlichen auf seiner internen Wissensbasis und der Fähigkeit, daraus Schlußfolgerungen zu ziehen, sowie der Fähigkeit zu lernen und sich Änderungen der Umwelt anzupassen.

4.2.4 Autonomes Handeln

Agenten sollten weitgehend autonom, d.h. ohne unnötige Interaktion mit der Umwelt, agieren können, da sie einem etwaigen Benutzer Arbeit abnehmen und nicht viele Handlungsentscheidungen von ihm treffen lassen sollen.

4.2.5 Kommunikation/Kooperation

Kommunikation bezeichnet die Fähigkeit eines Agenten, mit seiner Umwelt Informationen auszutauschen. Wird die Kommunikation auf einer höheren Ebene betrieben, und kann durch diese Interaktion mit der Umwelt gemeinsam eine Aufgabe gelöst werden, so wird auch von Kooperation gesprochen.

4.2.6 Mobilität

Mobilität ist die Fähigkeit eines Agenten, sich selbst innerhalb eines Netzwerkes zu bewegen.

4.2.7 Charakter

Insbesondere für Agenten, die über ein hohes Maß an Interaktion mit realen Personen verfügen, ist es sinnvoll, diesen nicht nur ein menschliches Erscheinungsbild zu geben, sondern dies auch durch menschliche Charaktereigenschaften und Kommunikation von Emotionen zu unterstreichen.

4.3 Einflußgebiete

Die oben aufgeführten Charakteristika spiegeln wieder, daß die Agententechnologie Ergebnisse aus vielen verschiedenen Forschungsgebieten nutzt und auch benötigt, um sich weiterzuentwickeln. Dies wird in der folgenden Abbildung verdeutlicht:

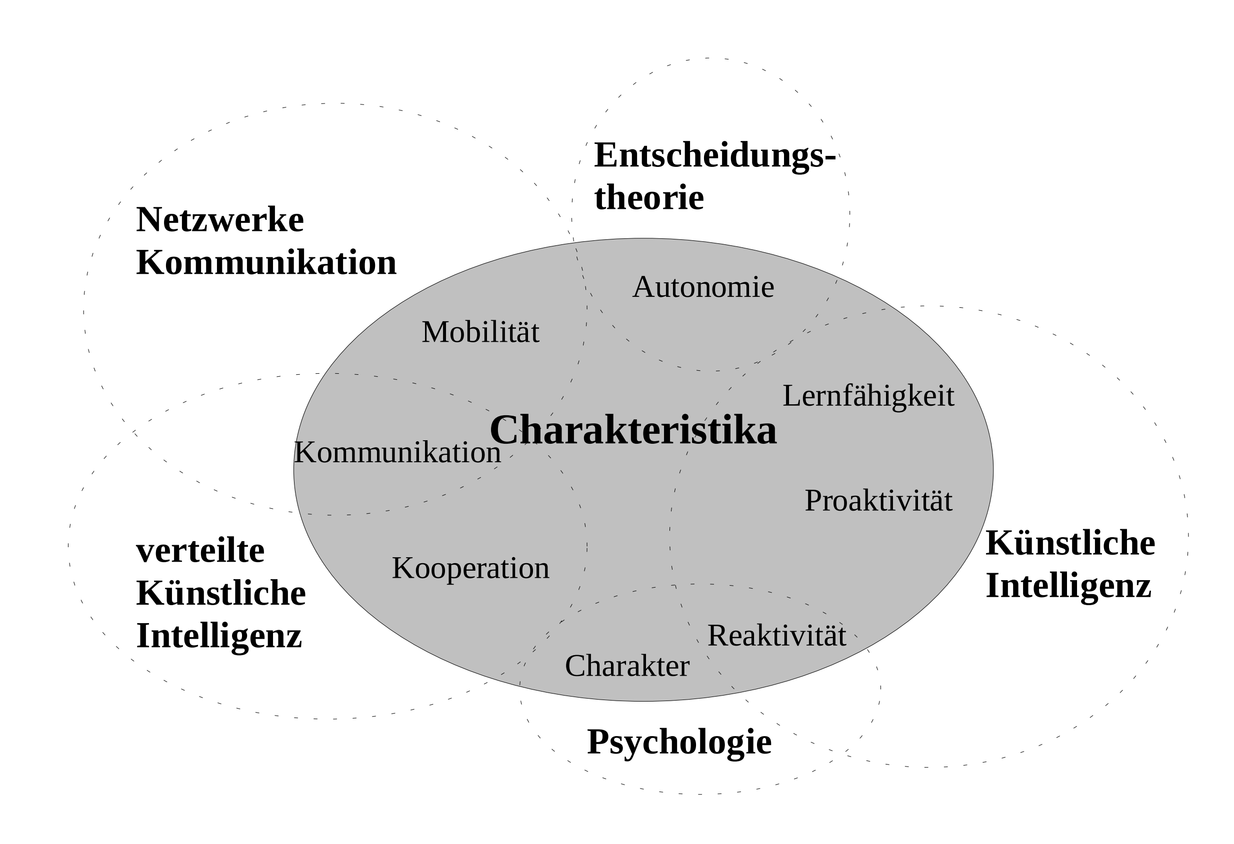


Abbildung 11 Charakteristika und deren Einflußgebiete

Als wichtigste Einflußgebiete und maßgeblich für die Entwicklung der Agententechnologie sind die Bereiche der Künstlichen Intelligenz, der verteilten Künstlichen Intelligenz und der Netzwerk- und Kommunikationssysteme.

4.4 Allgemeiner Aufbau eines intelligenten Agenten

Das Verhalten eines intelligenten Agenten läßt sich abstrakt so betrachten, daß er basierend auf einer Eingabe eine Verarbeitung durchführt und dazu eine Ausgabe produziert. Dieses bekannte EVA-Prinzip ist für Agenten nur auf einer sehr niedrigen Ebene anwendbar. Die Eingabe ist die Wahrnehmung der Umwelt des Agenten, es folgt eine intelligente Verarbeitung dieser Eindrücke und führt schließlich zu Aktionen des Agenten.

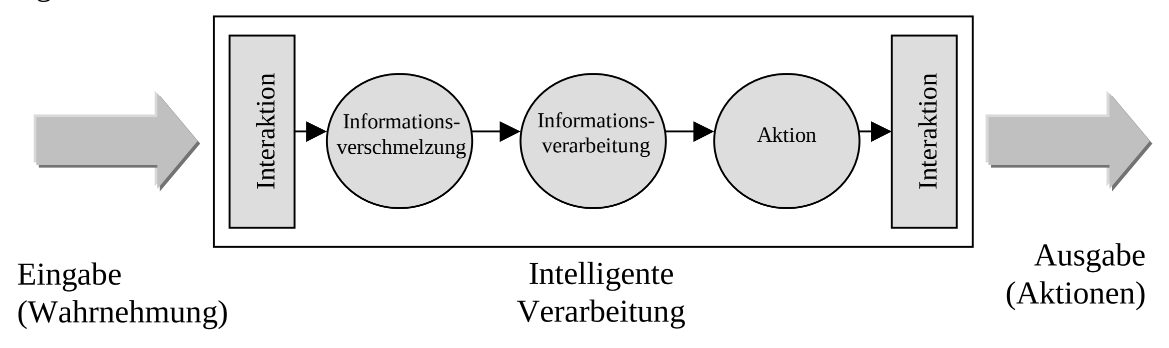


Abbildung 12 Arbeitsprozesse eines intelligenten Agenten

Wird dieser Arbeitsprozeß bei Agenten genauer betrachtet, so besitzt ein Agent zwei Interaktionsmodule. Mit dem einen nimmt er seine Umwelt, d.h. Benutzereingaben, andere Agenten, Datenbanken oder ähnliche Informationsquellen, wahr. Es liest also Informationen ein. Das andere dient dazu, sein Ergebnis zu kommunizieren, bzw. zu agieren. Nach der Wahrnehmung werden die eingehenden Informationen mit den schon vorhandenen in der Wissensbasis verschmolzen, um ein einheitliches Bild der Umwelt zu erlangen. Nun werden die so erhaltenen Informationen weiterverarbeitet, um sie zu interpretieren und damit Schlüsse für das weitere Vorgehen des Agenten zu erhalten. Dies wird benutzt, um die weiteren Aktionen des Agenten planen zu können. Entscheidet sich der Agent aufgrund der vorhandenen Informationen zu einer Aktion mit seiner Umwelt, so wird diese im nächsten Schritt durchgeführt. Eine Aktion könnte z.B. die Präsentation eines Ergebnisses für den Benutzer sein oder die weitere Interaktion mit einem anderen Agenten.

5 Maschinelle Lernverfahren

5.1 Einführung

Das Lernen von natürlicher Sprache ist seit Jahrzehnten ein intensiv untersuchtes Forschungsgebiet, an dem sich diverse Disziplinen beteiligt haben. Neben der Psychologie (Erforschung kindlicher Sprachentwicklung), der Linguistik (Vergleich unterschiedlicher Sprachen zur Ermittlung universeller Eigenschaften), der theoretischen Informatik (formale Betrachtung von Sprachen unter dem Gesichtspunkt ihrer Ausdruckskraft und Komplexität) und der künstlichen Intelligenz (Anwendungen natürlicher Sprachverarbeitung bei der Kommunikation intelligenter Systeme, z. B. in der Robotik) gab es vor allem in den letzten zwei Jahrzehnten wichtige Ergebnisse aus dem Bereich des Maschinellen Lernens. Der Schwerpunkt der Forschung in Bezug auf das Lernen von Sprache liegt hierbei vor allem auf der Analyse ihrer Syntax.

Dieses Kapitel beginnt mit einer Einführung in das induktive Lernen zur Einordnung des Systems in das Gebiet maschineller Lernverfahren. Danach folgt der thematische Schwerpunkt dieses Kapitels, die Grammatik-Inferenz.

Sowohl das einführende Kapitel zum induktiven Lernen als auch die Ausführungen zur Grammatik-Inferenz erfolgen vor dem Hintergrund des Paradigmas der Identifikation im Grenzwert. So werden bei der nachfolgenden Beschreibung entsprechender Lernverfahren zur Grammatik-Inferenz heuristische Verfahren, z. B. auf der Grundlage konnektionistischer bzw. stochastischer Modelle nicht diskutiert.

5.2 Induktives Lernen

Eine allgemeine Definition des induktiven Lernens stammt von [Angluin et al. 1983] und bezeichnet dieses als „...the act, process or result of an instance of reasoning from a part to a whole, from particulars to generals or from individuals to the universal“.

[Dupont 1997] definiert induktives Lernen im Kontext der Grammatik-Inferenz als „...the task of discovering common structures in examples which are supposed to be generated by the same process“.

Bei den logischen Schlußfolgerungsverfahren werden neben dem induktiven noch der deduktive und der abduktive Schluß betrachtet [Morik 1999]. Dem deduktiven Schluß kommt dabei eine besondere Bedeutung zu, da dieser innerhalb des induktiven Lernvorganges im Rahmen des Hypothesentests angewandt wird.

Unter Deduktion versteht man die formale Schlußfolgerungsmethode in der Logik. Mit dem deduktiven Schluß können aus wahren Aussagen wiederum wahre abgeleitet werden. Diese Ableitung wird durch sogenannte Schlußregeln realisiert, die die

Eigenschaften der Vollständigkeit und der Korrektheit erfüllen sollen. Die wichtigsten Schlußregeln sind zum einen der Modus Ponens $\frac{A, A \to B}{B}$ sowie zum anderen die Instantiierung $\frac{\forall x P(x)}{P(a)}$, wobei gilt, daß, falls alle Aussagen oberhalb der Linie wahr sind, dies auch für alle Aussagen unterhalb der Linie gilt.

Unter dem abduktiven Schluß wird im einfachsten Fall die Umkehrung des Modus Ponens verstanden und die entsprechende Schlußregel folgendermaßen formalisiert: $\frac{B, A \to B}{A}$.

Der abduktive Schluß ist, ebenso wie der induktive, nicht wahrheitserhaltend.

5.2.1 Spezifikation des Inferenzproblems

Von [Angluin et al. 1983] stammt eine allgemeine Beschreibung des induktiven Inferenzproblems. Die nachfolgende Spezifikation stellt auch den Rahmen für das später diskutierte Problem der Grammatik-Inferenz dar.

Folgende Punkte gehören zu einer Beschreibung des Inferenzproblems:

Die Klasse der zu lernenden Regeln (Ziel-Hypothesen)

Der Hypothesenraum (L_H)

Die Menge der Beispiele (L_E)

Die betrachtete Inferenzmethode

Ein Kriterium zur Evaluation des Lernerfolges der Inferenzmethode

Für den Hypothesenraum (L_H) muß gelten, daß jede zu lernende Regel von dem Hypothesenraum abgedeckt wird bzw. aus dem Hypothesenraum erzeugt werden kann. Die Menge der Ziel-Hypothesen sollte dabei eine Teilmenge von L_H sein.

Unter der Inferenzmethode soll im folgenden ein Lernverfahren verstanden werden, das den eigentlichen induktiven Lernschritt (die Hypothesengenerierung) mit einem nachfolgenden deduktiven Schritt (dem Hypothesentest) verbindet. Beim Hypothesentest wird dabei die Konsistenz der Hypothesen mit der Beispielmenge geprüft.

Unter den Kriterien des Lernerfolges wird z. B. das in Kapitel 5.2.6 beschriebene Paradigma der Identifikation im Grenzwert verstanden.

[Pitt 1989] führt in seiner Definition des Inferenzproblems einen weiteren Punkt ein. Hierbei stehen zusätzliche Informationen über die gesuchten Regeln zur Verfügung. Diese zusätzlichen Informationen können z. B. in Form eines *knowledgeable teachers* gegeben sein, der Anfragen (*Queries*) in Bezug auf die gesuchte Hypothese beantworten kann.

Die Beschreibung eines Inferenzproblems soll anhand der Inferenz von regulären Sprachen aus positiven und negativen Beispielen verdeutlicht werden:

Die Klasse der zu lernenden Regeln sei die Menge der regulären Ausrücke über einem vorgegebenen Alphabet. Der Hypothesenraum sei die Menge aller DFA (denkbar wäre z. B. auch die Menge der regulären Sprachen bzw. der nichtdeterministischen Automaten). Die Menge der Beispiele besteht aus Tupeln (b,k), wobei $k \in \{0,1\}$ angibt, ob ein String b Element der regulären Sprache L ist. Die Inferenzmethode zur Induktion regulärer Sprachen aus klassifizierten Beispielen sei RPNI (siehe Kapitel 5.3.5.2.1). Das Kriterium der Identifikation regulärer Sprachen im Grenzwert ist im vorliegenden Fall erfüllt.

5.2.2 Vergleich von Hypothesen

Eine Möglichkeit, die „Qualität" der Hypothesen, und damit die Qualität des induktiven Lernergebnisses zu bewerten, ist es, die Einfachheit (*simplicity*) dieser Hypothesen zu betrachten. Dabei ist unter Einfachheit die Komplexität der Hypothesen, bzw. die Größe des Hypothesenraumes zu verstehen.

Im Fall der Inferenz endlicher Automaten wird als Kriterium der Einfachheit z. B. die Anzahl der Zustände des erzeugten Automaten gewählt. Diesem Kriterium liegt die Erkenntnis zugrunde, daß die Generalisierungsgenauigkeit bei Automaten mit einer geringen Anzahl von Zuständen höher ist als bei Automaten mit einer höheren Anzahl von Zuständen. Weitere Maße sind je nach Art des Hypothesenraumes denkbar, so z. B. Inklusionsbeziehungen von Mengen bei der Inferenz von Sprachen etc.

Zur Bewertung der Hypothesen kann eine partielle Ordnung über der Menge der Hypothesen bezüglich des gewählten Maßes definiert werden. Die Hypothesen, also die Elemente dieser Ordnung, werden gemäß dieses festgelegten Maßes sortiert. In jedem Schritt des Inferenzprozesses, also nach jeder neuen Eingabe, wird diese Liste aktualisiert, indem die beste Beschreibung für die Eingabedaten ermittelt wird. Dabei muß die Voraussetzung gelten, daß die Konsistenz mit der Beispielmenge erhalten bleibt.

5.2.3 Eigenschaften von Inferenzmethoden

Grob gesagt ist eine Inferenzmethode ein „...computable process of some kind that reads in examples and outputs guesses from the hypothesis space."[Angluin et al. 1983].

Zum besseren Verständnis induktiver Inferenzverfahren sollen im folgenden die wichtigsten Charakteristika solcher Verfahren kurz definiert werden.

Grundlage der folgenden Begriffsbildungen sind jeweils ein Kriterium zur Evaluation der korrekten Inferenz C, eine Datenrepräsentation D und ein Inferenzverfahren I.

5.2.3.1 Ein Rahmen zur Evaluation von Inferenzverfahren

Zur Erfolgskontrolle eines Inferenzverfahrens gibt es eine Reihe von Kriterien. Die folgenden Begriffe stellen dabei eine definitorische Grundlage aller Inferenzverfahren dar. Kapitel 5.2.6 beschreibt mit der Identifikation bzw. Enumeration im Grenzwert wichtige Evaluationsmethoden.

Der *Scope* bezeichnet die Menge der Regeln, die ein konkretes Verfahren M für ein bestimmtes D inferiert. Ein Verfahren M_1 ist mächtiger (*more powerful*) als ein Verfahren M_2, falls der Scope von M_2 in M_1 enthalten ist. Eine Menge von Regeln U ist inferierbar in Bezug auf die Methode M, falls U im Scope von M enthalten ist.

Die Dateneffizienz einer Inferenzmethode M ist definiert als die Menge von Daten, die benötigt wird, um eine korrekte Hypothese zu inferieren. Gegeben ist also eine unendliche Folge von Beispielen, so daß jede Regel aus der Regelmenge durch mindestens ein Beispiel abgedeckt wird (*admissible presentation*). Der Konvergenzpunkt ist nun der kleinste Index i der Eingabedaten x_n mit $i \leq n$, so daß M die gesuchte Hypothese ausgibt und alle nachfolgenden Eingabedaten x_j mit $j>i$ zu keiner Veränderung der korrekten Hypothese mehr führen. Ein Verfahren M_1 besitzt eine höhere Dateneffizienz als ein Verfahren M_2, falls M_1 dieselbe Menge der Regeln identifiziert wie M_2, unabhängig von der zulässigen Folge der Eingabedaten, und der Konvergenzpunkt von M_1 kleiner ist als der von M_2.

Neben dem Konvergenzpunkt und der damit verbundenen Dateneffizienz ist die Anzahl der *mind changes*, also der Häufigkeit, mit der Hypothesen im Laufe des Inferenzprozesses verändert werden, ein wichtiges Kriterium.

5.2.4 Einschränkungen von Inferenzmethoden

Die Konsistenz (*consistence*) verlangt von einer Methode M, daß zu jedem Zeitpunkt der Inferenz die Hypothesen konsistent sind mit den gegebenen Eingabedaten, d. h. daß sie alle positiven und kein negatives Beispiel abdecken.

Die Verläßlichkeit (*reliability*) besagt, daß im Falle einer Konvergenz der Methode M die korrekte Hypothese inferiert wird. Im negativen Fall ist die Anzahl der unterschiedlichen Hypothesen unendlich.

Eine Methode ist konservativ (*conservative*), falls sich die Hypothese bei neuen Eingabedaten nur dann ändert, wenn sich eine Inkonsistenz der Hypothese mit den Eingabedaten ergeben hat.

5.2.5 Praktische Inferenzmethoden

Das Verfahren, Hypothesen innerhalb des Suchraumes linear anzuordnen, ist denkbar ineffizient und wird heute in der Regel durch Verfahren ersetzt, die sich durch eine Strukturierung des Hypothesenraumes auszeichnen. Durch diese Strukturierung ist es möglich, im Falle einer Inkonsistenz mehr als nur jeweils die letzte Hypothese auszuschließen.

Dieser Ansatz der Strukturierung des Hypothesenraumes zur effizienteren Inferenz von Hypothesen ist einer der grundlegenden Ansätze zahlreicher Verfahren der Grammatik-Inferenz, insbesondere bei regulären Sprachen. Der Ansatz soll an dieser Stelle nur skizziert werden - eine ausführlichere Behandlung erfolgt im Zusammenhang mit der DFA-Generierung im Kapitel 8. Zunächst wird ein DFA erzeugt, der konsistent ist mit der Menge der positiven und negativen Beispiele. Dieser Automat wird durch sukzessive Zusammenfassung (*Merging*) von Zuständen komprimiert. Der Hypothesenraum, der durch die so entstandenen endlichen Automaten gebildet wird, stellt eine erhebliche Einschränkung des - ursprünglich unstrukturierten Suchraumes dar.

Ein anderer Ansatz ist die Betrachtung von Subsumptionsbeziehungen. Hierbei geht man von einer partiellen Ordnung des Hypothesenraumes in Bezug auf die Generalität / Spezifität der Hypothesen aus. Sei H_1 eine Hypothese, die ein negatives Beispiel abdeckt. Dann gibt es keine generellere Hypothese H_2, die konsistent ist mit der gegebenen Beispielmenge. Sei nun H_1 eine Hypothese, die ein positives Beispiel nicht abdeckt. Dann sind alle spezielleren Hypothesen (als H_1) ebenfalls inkonsistent mit der Beispielmenge und müssen nicht weiter untersucht werden.

Das Prinzip der (Halb-)Ordnung eines Suchraumes unter Ausnutzung von Subsumptionsbeziehungen machte sich [Mitchell 1997] im Rahmen seines Versionenraum-Modells zu nutze. Dieser Versionenraum enthält zu Beginn nur die Menge der generellsten Hypothesen G, die kein negatives Beispiel abdecken und die Menge der speziellsten Hypothesen S, die alle positiven Beispiele abdecken. Im Laufe des Inferenzprozesses verkleinert sich der Versionenraum kontinuierlich, bis die Mengen S und G in einem Element, der gesuchten Hypothese, zusammenfallen.

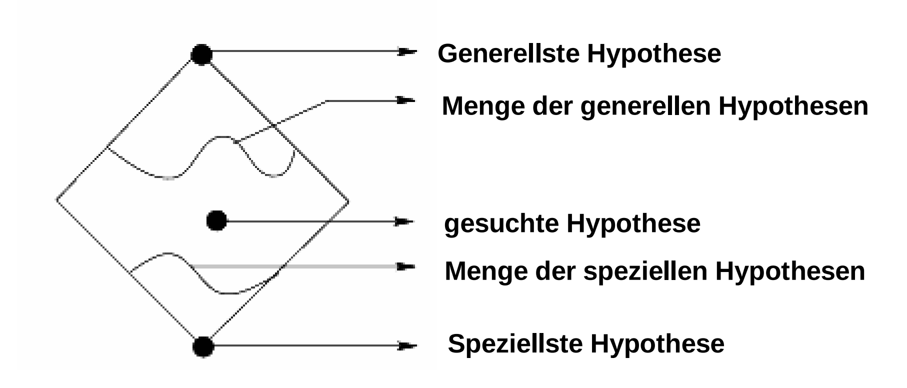


Abbildung 13 Versionenraum-Modell

Beim sogenannten *Pruning* wird im Fall der Inkonsistenz einer Hypothese mit der Beispielmenge nach dem Grund für diese Inkonsistenz, dem sogenannten *forbidden*

feature gesucht. Bei der nachfolgenden Suche werden Hypothesen mit dieser Eigenschaft ausgeschlossen.

Die *Hill-climbing-Methode* (ausführlichere Beschreibung der Methode z. B. in [Morik 1997]) ermittelt die Hypothese, die bezogen auf eine gegebene Metrik ein lokales Maximum im Hypothesenraum annimmt. Eine sinnvolle Metrik M im Rahmen der Grammatik-Inferenz könnte die Komplexität einer Grammatik G und die Übereinstimmung der erzeugten Sprache L(G) mit der gegebenen Beispielmenge S in Beziehung setzen.

5.2.6 Kriterien zur Evaluation von Inferenzmethoden

Die folgenden Verfahren von [Gold 1967] zur Identifikation bzw. Enumeration, d. h. zur exakten Bestimmung einer Sprache, dienen der Erfolgskontrolle des induktiven Inferenzprozesses.

5.2.6.1 Identifikation im Grenzwert (identification in the limit)

Gegeben ist eine Klasse von Sprachen. Die einzelnen Sprachen sind Strings über einem gemeinsamen Alphabet. Des weiteren gibt ein Lehrer (*teacher*) die Form vor (*method of information presentation*), in der die Beispiele dem Schüler (*learner*) präsentiert werden. Man unterscheidet dabei den Typ „text“, unter dem vereinfacht gesagt eine Präsentation der positiven Beispiele der gesuchten Sprache verstanden wird, sowie den Typ „informant“, der eine Klassifizierung der gegebenen Informationen in positive und negative Beispiele der Sprache liefert. Zur eindeutigen Benennung der vom Schüler erkannten Sprache steht diesem eine sogenannte *naming relation* zur Verfügung.

Der Lehrer wählt nun eine Sprache aus der festgelegten Sprachklasse und eine Methode der Präsentation der Beispiele aus. Nach jedem Beispiel gibt der Schüler aufgrund der bisher vom Lehrer erhaltenen Informationen einen Vorschlag in Bezug auf die zu identifizierende Sprache ab. Dieser Prozeß ist prinzipiell unendlich.

Eine Sprachklasse wird als identifiziert bezeichnet, falls es einen Algorithmus gibt, so daß nach einem Zeitpunkt t alle nachfolgenden Hypothesen des Schülers identisch und korrekt sind.

Abbildung 14 veranschaulicht diesen Prozeß vom Blickwinkel der entsprechenden Grammatiken einer Sprache. Grammatik G_0 ist die zu identifizierende Grammatik. Der Lehrer erzeugt jeweils ein Beispiel aus der gesuchten Sprache und präsentiert es dem Schüler. Dieser schlägt daraufhin eine bestimmte Grammatik G_i vor. Die Grammatik G_* stellt im Fall der Identifikation eine Repräsentation von $L(G_0)$ dar.

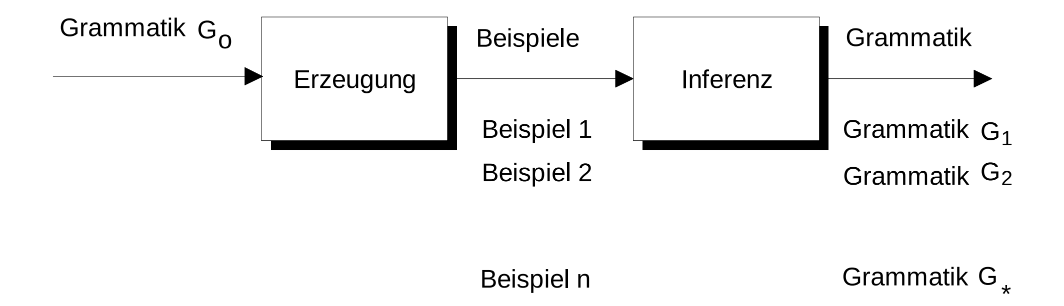


Abbildung 14 Identifikation formaler Grammatiken im Grenzwert

Abbildung 15 verdeutlicht folgende Erkenntnisse in Bezug auf das Lernen von Sprache: Zum einen sind mit positiven Beispielen allein (in der Abbildung als „text“ deklariert) nur die Sprachen endlicher Kardinalität erlernbar. Alle anderen Sprachen (auch die regulären) sind somit nicht allein durch positive Beispiele lernbar. Zum anderen sind mit klassifizierten Beispielen (in der Abbildung als „informant“ bezeichnet) alle Sprachklassen unterhalb der rekursiven Sprachen identifizierbar.

Method of presentation	Class of languages
	Recursively enumerable
	Recursive
Informant ———	Primitive recursive
	Context-sensitive
	Context-free
	Regular
Text ———	Finite cardinality

Abbildung 15 Lernbarkeit formaler Sprachen nach Gold

Trotz der grundlegenden Bedeutung von Golds Resultaten auf theoretischer Ebene sind die Ergebnisse für die praktische Anwendung des induktiven Lernens, insbesondere der Grammatik-Inferenz, von eher eingeschränkter Bedeutung. So kritisiert [Finch 1993] Golds Paradigma in folgenden Punkten: Zum einen sei das Kriterium des Lernerfolges, die Identifikation *aller* Sätze einer Sprache, zu strikt, da eine gute Approximation einer Sprache in vielen Fällen ausreicht („...It is not much more useful to be able to identify all sentences than 99.99% of the sentences.“). Zum anderen ist eine implizite Voraussetzung für die Korrektheit der Identifikation einer Sprache nach Gold, daß die gegebenen Beispiele fehlerfrei (*noise-free*) sind, d. h. vom Lehrer korrekt klassifiziert wurden. Andernfalls wird die fehlerhafte Eingabe dazu führen, daß der Algorithmus eine inkorrekte Grammatik lernt. Ferner kritisiert Finch, daß sowohl durch die potentiell unendliche Laufzeit des Lernprozesses als auch durch die Größe des Hypothesenraumes

unrealistische Rahmenbedingungen in Bezug auf die praktische Anwendung gesetzt wurden.

5.2.6.2 Identifikation durch Aufzählung (identification by enumeration)

Bei diesem Verfahren werden die gesamten Elemente des Hypothesenraumes systematisch aufgezählt. Nach jedem neuen Beispiel wird diese Aufzählung durchsucht und das erste Element dieser Liste (z. B. die gefundene Sprache bzw. der gefundene DFA) ausgegeben, das mit den bisherigen Daten konsistent ist. Dabei muß natürlich gewährleistet sein, daß die Erzeugung der Aufzählung und die Konsistenzprüfung mit den aktuellen Daten berechenbar ist.

Der größte Nachteil dieser Methode ist, daß der Hypothesenraum, der jeweils durchsucht werden muß, exponentiell in der Größe der Eingabemenge ist.

5.3 Grammatik-Inferenz

Die Grammatik-Inferenz kann als eine Instanz des induktiven Lernens betrachtet werden. Sie stellt im Rahmen der natürlichen Sprachverarbeitung ein Werkzeug dar, mit dem Strukturen innerhalb einer Sprache - genauer innerhalb der Syntax einer Sprache - ermittelt werden können.

Definiert ist die Grammatik-Inferenz als ein Prozeß, der aus einer Menge von Beispielen die erzeugende Grammatik lernt. Neben der Menge der positiven Beispiele, also der grammatikalisch korrekten Sätze, werden in einigen Algorithmen auch negative Beispiele, grammatikalisch inkorrekte Sätze, zur Inferenz der Grammatik verwendet.

Als grundlegende Einführungen in den Bereich der Grammatik-Inferenz sei an dieser Stelle z. B. auf [Parekh et al. 1998] und [Dupont 1997] verwiesen.

5.3.1 Definitionen

Eine formale Grammatik ist ein 4-Tupel $G=(V_t,V_n,S,P)$, wobei V_t die Menge der terminalen Symbole, V_n die Menge der nicht-terminalen Symbole, S ($S \in V_n$) die Menge der Startsymbole und P die Regelmenge der Form $\alpha \rightarrow \beta$ ist, wobei α und β Worte über dem Alphabet ($V_t \cup V_n$) sind und α mindestens ein Element aus V_n enthält.

Eine Sprache L(G) ist die Menge aller Worte w, die in endlich vielen Schritten aus den Produktionsregeln der Grammatik G abgeleitet werden kann (formal gilt: $L = \{w \mid S \rightarrow^* w\}$).

5.3.2 Chomsky-Hierarchie

Eine Strukturierung der Grammatiken in vier Klassen mit steigender Einschränkung der zugrundeliegenden Regelmengen und sinkendem Umfang der erzeugten Sprachen wird durch die Chomsky-Hierarchie für formale Sprachen gegeben (siehe dazu auch [Hopcroft et al. 1988]). Zu jeder Grammatik-Klasse existiert ein entsprechendes

Maschinenmodell zur Lösung des Wortproblems, also zur Beantwortung der Frage, ob ein beliebiges Wort Element der Sprache ist.

Chomsky-0-Grammatiken (uneingeschränkte Sprachen) enthalten nur Regeln der Form $\varphi \rightarrow \psi$ ohne weitere Einschränkungen.

Chomsky-1-Grammatiken (kontextsensitive Sprachen) enthalten Regeln der Form $c_1 A\, c_2 \rightarrow c_1 \omega\, c_2$ mit den Bedingungen $A \in V_N$, $\omega \neq \emptyset$ und ω, c_1, $c_2 \in V^*$.

Das zugehörige Maschinenmodell für die Klasse der kontextsensitiven Sprachen sind die nichtdeterministischen Turingmaschinen mit einseitig begrenztem Band. Für diese Klasse von Sprachen ist die Lösung des Wortproblems NP-vollständig.

Chomsky-2-Grammatiken (kontextfreie Sprachen) enthalten folgende Regeln : $A \rightarrow \omega$ mit den Bedingungen $A \in V_N$, $\omega \in V^*$ und $\omega \neq \emptyset$.

Obwohl es zur Lösung des Wortproblems effiziente Algorithmen gibt (z. B. Cocke-Younger-Casami, bzw. Earley) ist in der Klasse der kontextfreien Grammatiken die Frage der Teilmengenbeziehung zweier Sprachen ($L(G_1) \supseteq L(G_2)$) sowie die Frage nach der Schnittmenge zweier Sprachen ($L(G_1) \cap L(G_2)$) unentscheidbar. Dies hat zur Folge, daß viele Lernalgorithmen für kontextfreie Sprachen heuristischer Natur sind. Das korrespondierende Maschinenmodell für kontextfreie Sprachen sind die nondeterministic pushdown automata (PDA).

Chomsky-3-Grammatiken (reguläre Sprachen) enthalten folgende Regeln : $A \rightarrow aB$ und $A \rightarrow a$ mit $a \in V_T$ und $A,B \in V_N$

Die Menge der regulären Sprachen mit dem korrespondierenden Maschinenmodell, der Klasse der DFA (deterministischen endlichen Automaten) ist die am besten untersuchte und insbesondere im Hinblick auf die Grammatik-Inferenz am häufigsten benutzte Klasse der formalen Sprachen. Als Repräsentationsformalismus für reguläre Grammatiken stehen reguläre Ausdrücke, Produktionsregeln, DFA und NFA (nichtdeterministische endliche Automaten) zur Verfügung. Am häufigsten wurde dabei das Modell der deterministischen endlichen Automaten verwendet. Die Gründe hierfür liegen unter anderem darin, daß für einige häufig benutzte Operationen im Rahmen der Grammatik-Inferenz, wie z. B. der Minimierung von Automaten, dem Test auf Äquivalenz, bzw. Inklusionsbeziehungen zweier Sprachen effiziente Algorithmen existieren.

5.3.3 Spezifikation der Grammatik-Inferenz

Das allen Methoden der Grammatik-Inferenz zugrundeliegende Prinzip wird in Abbildung 16 verdeutlicht. Gesucht wird eine Grammatik G (das Ergebnis des Inferenz-Prozesses), die die zu lernende Grammatik G_0 identifiziert, bzw. möglichst gut approximiert. Dabei ist die Existenz der Grammatik G_0 eine reine Modellierungshypothese. Anders formuliert kann man die Grammatik G_0 als Black-Box betrachten, von der nur das Verhalten (die von G_0 erzeugten Sätze, bzw. Beispiele)

bekannt ist. Die Art der Eingabedaten kann jedoch je nach Verfahren variieren. So stellen einige Verfahren besondere Anforderungen an die Beispielmenge (z. B. explizite negative Beispiele), bzw. benötigen neben den Beispielen zusätzliche Informationen. Dieser Aspekt der Inferenz wird detaillierter in Kapitel 5.3.4 behandelt.

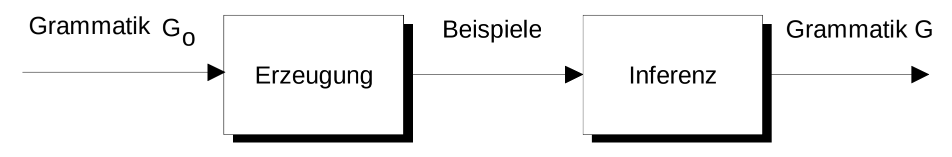


Abbildung 16 Prinzip der Grammatik-Inferenz

5.3.4 Theoretische Resultate der Grammatik-Inferenz regulärer Sprachen

Gold legte mit den folgenden Ergebnissen aus dem Bereich der Identifikation formaler Sprachen einen Grundstein der theoretischen Forschung im Bereich der Grammatik-Inferenz. Er bewies, daß sowohl die Identifikation regulärer Sprachen aus positiven Beispielen allein [Gold 1967], als auch die Identifikation eines minimalen Automaten, der konsistent ist mit einer *beliebigen* Menge positiver und negativer Beispiele [Gold 1978] nicht effizient berechenbar ist.

Das Problem, eine Grammatik eindeutig aus der Menge der positiven Beispiele zu bestimmen bzw. zu identifizieren wird deutlich, wenn man bedenkt, daß keine 1:1-Beziehung zwischen einer Sprache L(G) und einer erzeugenden Grammatik G existiert. Das bedeutet, daß jede endliche Beispielmenge S von einer unendlichen Zahl von Grammatiken erzeugt werden kann. Ohne weitere Information, sei es z. B. durch zusätzliche negative Beispiele oder die Möglichkeit Anfragen über die Form der Ziel-Grammatik zu stellen ist eine eindeutige Zuordnung somit nicht möglich. Siehe hierzu auch [Fu et al. 1975].

Diese zunächst ernüchternden Resultate leiteten eine Reihe von Forschungen nach effizienteren Inferenz-Methoden ein.

Dabei entstanden zwei Ansätze zur effizienten Identifikation regulärer Sprachen. Zum einen wurden bestimmte Anforderungen an die Form der Beispielmenge gestellt, zum Anderen wurden neben den Beispielen zusätzliche Information bereitgestellt, die die Lernaufgabe erleichtern sollten.

[Trakhtenbrot et al. 1973] gehen in ihrem Algorithmus von einer vollständigen klassifizierten Beispielmenge aus, d.h. einer vollständigen Aufzählung der Beispiele bis zu einer bestimmten Länge und einer Funktion c, die die jeweiligen Beispiele auf die Menge der positiven und negativen Beispiele abbildet.

Der Algorithmus von [Oncina et al. 1992] benötigt hingegen keine negativen Beispiele. Grundlage dieses Verfahrens ist die sogenannte charakteristische Menge, die Informationen über die Zustände und Transitionen des Ziel-Automaten beinhaltet (für eine formale Definition der charakteristischen Menge siehe Kapitel 8.1). Die

Identifikation des Ziel-Automaten wird dadurch gewährleistet, daß die gegebenen Beispiele diese charakteristische Menge enthält.

Eine bereits erwähnte Möglichkeit, zusätzliche Information zu erhalten ist durch sogenannte *membership*, bzw. *equivalence queries* gegeben. Eine membership querie befragt ein Orakel nach der Klassifikation für ein Beispiel x. Im Fall der Inferenz endlicher Automaten könnte eine solche Anfrage lauten :“Wird der Satz x von der Sprache L erzeugt ?“, bzw. alternativ „Wird der Satz x vom DFA akzeptiert ?“.Eine equivalence querie in Bezug auf eine gegebene Hypothese H wäre z. B. eine Frage der Form „Ist die gegebene Hypothese H äquivalent mit der gesuchten Hypothese H*“. Das Orakel antwortet, wiederum vor dem Hintergrund endlicher Automaten mit „ja“ falls der aktuelle Automat M äquivalent ist mit dem gesuchten Automaten M*, andernfalls antwortet er mit „nein“ und liefert einen String w zurück, der von M, nicht aber von M* akzeptiert wird. Formal gilt für den String $w \in L(M) \oplus L(M^*)$.

Des weiteren wurden eine Reihe von Subklassen regulärer Sprachen untersucht (z. B. die Klassen der k-kontextuellen bzw. der k-reversiblen Sprachen). Diese Klassen unterliegen nicht mehr den o. g. Einschränkungen, und sind für viele Anwendungen noch ausdrucksstark genug. Der im vorliegenden System gewählte Ansatz geht bei der Inferenz im übrigen von einer solchen Teilmenge der regulären Sprachen, konkret von den k-reversiblen Sprachen, aus.

5.3.5 Lernalgorithmen für reguläre Sprachen

Zur bessern Einordnung der im System verwendeten Methoden in die Klasse der Lernverfahren für reguläre Sprachen gibt Abbildung 17 einen Einblick in die Klasse der exakten Lernmethoden. Die Ausführungen beschränken sich dabei auf einen sehr begrenzten, für die Einordnung relevanten Ausschnitt aus der Vielzahl der Methoden. Auf die Beschreibung heuristischer Verfahren zur Grammatik-Inferenz, wie zum Beispiel auf der Grundlage eines konnektionistischen, bzw. stochastischen Ansatzes oder Verfahren auf der Basis genetischer Algorithmen wird in diesem Zusammenhang verzichtet.

Alle nachfolgend beschriebenen Verfahren basieren auf dem Paradigma der Identifikation im Grenzwert und die betrachtete Hypothesensprache ist die Klasse der endlichen Automaten.

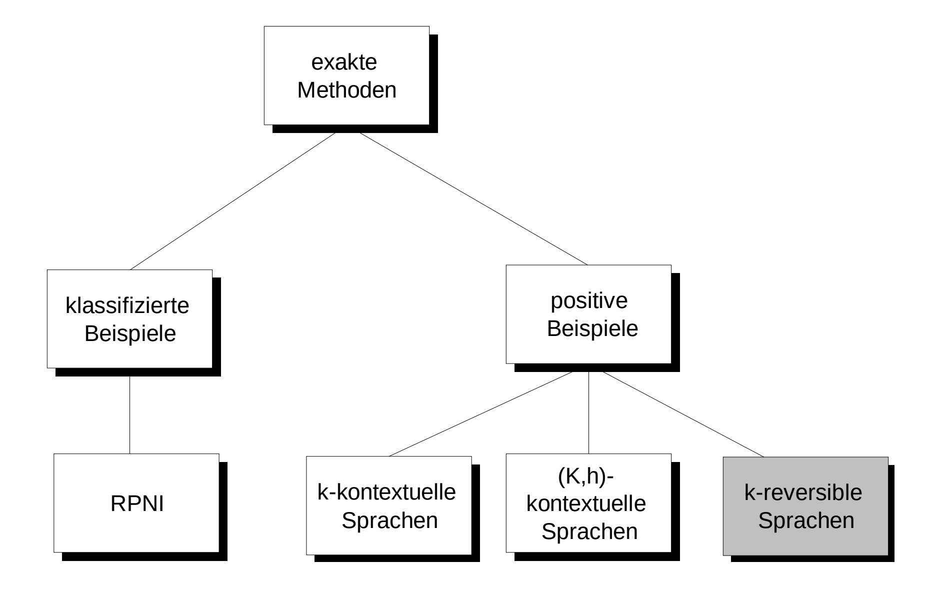


Abbildung 17 Hierarchie der Lernverfahren für reguläre Sprachen

Charakteristisch für die Klasse der exakten Methoden ist eine garantierte Konvergenz in Hinblick auf die gesuchte Grammatik unter der Voraussetzung, daß eine widerspruchs- und fehlerfreie, sowie repräsentative Beispielmenge existiert.

Innerhalb der Klasse der exakten Methoden ist das Vorhandensein negativer Beispiele ein wichtiges Unterscheidungskriterium. Aus diesem Grund werden sowohl Methoden diskutiert, bei denen ausschließlich positive Beispiele zur Verfügung stehen (wie z. B. der im System verwendeten Methode), als auch ein Verfahren, das mit vollständigen klassifizierten Beispielmengen arbeitet.

Im Gegensatz zu Inferenzverfahren für kontextfreie Grammatiken gibt es im Rahmen regulärer Sprachen eine Reihe von Verfahren, die unter den bereits diskutierten Voraussetzungen eine Identifikation der gesuchten Grammatik gewährleisten.

5.3.5.1 Methoden ohne negative Beispiele

Wie bereits erwähnt, ist das Problem der Identifikation einer Grammatik allein auf der auf der Grundlage positiver Beispiele nicht effizient berechenbar. Eine Möglichkeit dieses Problem zu umgehen ist es, sich auf Teilmengen regulärer Sprachen zu beschränken. Exemplarisch für diesen Ansatz sollen die Klassen der k-kontextuellen bzw. (k,h)-kontextuellen Sprachen genannt werden. In diese Klasse der exakten Methoden, die keine negativen Beispiele verwenden, fällt auch die in Abbildung 17 grau unterlegte Klasse der k-reversiblen Sprachen.

[Ahonen 1996] faßt die Gemeinsamkeiten dieser im folgenden betrachteten Sprachklassen wie folgt zusammen : “Our assumption is that the grammars used in structured documents have only limited context in the following sense. If a sufficiently long sequence of elements occur in two places in the examples, the elements that can follow this sequence are independent of the position of the sequence in the

document...The classes of k-reversible languages, k-contextual languages and (k,h)-contextual languages satisfy this condition in varying degrees".

Die im vorliegenden System verwendeten Lernverfahren basieren auf den Algorithmen von Angluin zur Inferenz k-reversibler Sprachen. Aus diesem Grund wird diese Sprachklasse ausführlich im Kapitel 8.3.3.1 diskutiert.

5.3.5.1.1 k-kontextuelle Sprachen

Die Intuition bei k-kontextuellen Sprachen ist, daß aus dem Auftreten zweier identischer String-Sequenzen der Länge k folgt, daß die nachfolgenden Elemente der Sequenz in beiden Fällen übereinstimmen.

[Muggleton 1990] gibt dazu folgende Definition der k-kontextuellen Sprachen. „A regular language is k-contextual for all strings u_1,u_2,w_1,w_2 and v, if and only if u_1vw_1 and u_2vw_2 are in L and |v| = k, then $T_L(u_1v)=T_L(u_2v)$.". [5]

Neben der formalen sprachlichen Beschreibung dieser Sprachklasse ist auch eine Deutung in der Terminologie der endlichen Automaten möglich.

Hier gilt, daß eine reguläre Sprache k-kontextuell ist „if and only if there is a finite automaton M such that L=L(M), and for any two states p_0 and q_0 of M and any string v with |v|=k we have: if there are states p_k and q_k of M such that $\delta (p_0,v)=p_k$ and $\delta(q_0,v)=q_k$, then $p_k=q_k$." [Ahonen 1996]

5.3.5.1.2 (k,h)-kontextuelle Sprachen

Die bereits für k-kontextuelle Sprachen formulierte Bedingung wird nun im Fall der (k,h)-kontextuellen Sprachen insoweit modifiziert, als daß das Auftreten zweier Sequenzen nun bedingt, daß die nachfolgenden Elemente schon nach h Worten übereinstimmen.

[Ahonen 1996] definiert die Klasse der (k,h)-kontextuellen Sprachen wie folgt: "Let $0 \leq h \leq k$. A regular language L is (k,h)-contextual if and only if for all strings u_1,u_2,w_1, and w_2, and input symbols $a_1,...,a_k$, if $u_1a_1...a_kw_1$ and $u_2a_1...a_kw_2$ are in L, then for every i, with $0 \leq h \leq i \leq k$ and each pair of strings v_1 and v_2 such that $v_1a_i...a_kw_1 \in L$ and $v_2a_i...a_kw_2 \in L$, we have $T_L(v_1a_i)=T_L(v_2a_i)$."

Zur Konstruktion eines k-kontextuellen, bzw. (k,h)-kontextuellen Automaten werden die Zustände des ursprünglichen Automaten solange zusammengefaßt, bis die jeweils formulierten Bedingungen nicht mehr erfüllt sind. Der entstandene Automat ist dann k-kontextuell, bzw. (k,h)-kontextuell.

[5] T_L ist die Menge aller Suffixe einer Sprache L. (siehe hierzu auch die Definitionen in Kapitel 8.1)

5.3.5.2 Methoden mit negativen Beispielen

5.3.5.2.1 RPNI

[Oncina et al. 1992] beschreibt mit RPNI (Regular Positive and Negative Inference) ein Verfahren, daß in polynomieller Zeit (genauer in $O((|S_+|+|S_-|)\cdot|S_+|^2))$ einen Automaten identifiziert, der konsistent ist mit der gegebenen Beispielmenge $S=S_+\cup S_-$. Dabei ist S_+ die Menge der positiven und S_- die Menge der negativen Beispiele.[6]

Des weiteren garantiert der Algorithmus, eine kanonische Repräsentation des Ziel-Automaten zurückzugeben, falls die charakteristische Menge dieses Automaten in S enthalten ist (für exakte Begriffsdefinitionen siehe Kapitel 8.1)

Der Algorithmus konstruiert zuerst eine Repräsentation der positiven Beispielmenge S_+ in Form eines sogenannten Präfixbaumes. Dieser spezielle DFA akzeptiert ausschließlich die Strings aus S_+ und keine sonst. Nun werden die Strings, die vom Präfixbaumes erzeugt werden in lexikographischer Ordnung sortiert. Jeder Zustand des Präfixbaumes wird nachfolgend gemäß der Position seines korrespondierenden Strings in der sortierten Liste numeriert. Der Algorithmus testet nun systematisch alle Zustandspaare gemäß dieser Ordnung, und faßt diese zu einem Zustand zusammen (generalisiert den bisherigen Automaten) falls der entstehende Automat noch konsistent ist mit der Beispielmenge S, d.h., insbesondere, daß er keine Beispiele aus S_- akzeptiert.

[6] Eine inkrementelle Erweiterung der RPNI (RPNI2) stellte im übrigen [Dupont 1996] in seiner Arbeit vor.

II Realisierung

6 Systemarchitektur

Nachdem in den bisherigen Kapiteln die Grundlagen im Bereich der angewandten maschinellen Lernverfahren, der intelligenten Softwareagenten und der radiologischen Befundung vorgestellt wurden soll dieses Kapitel einen Überblick über den konkreten Aufbau des entwickelten Systems geben.

Wie bereits in der Einleitung erwähnt, ist die Aufgabe dieses Systems, aus einer Menge strukturierter Texte (am Beispiel radiologischer Befundungstexte) eine adäquate interne Repräsentation zu erzeugen. Basierend auf dieser Repräsentation wird nun eine Benutzeroberfläche generiert, die es dem Arzt ermöglicht neue Befundungstexte effizient und strukturiert zu erstellen.

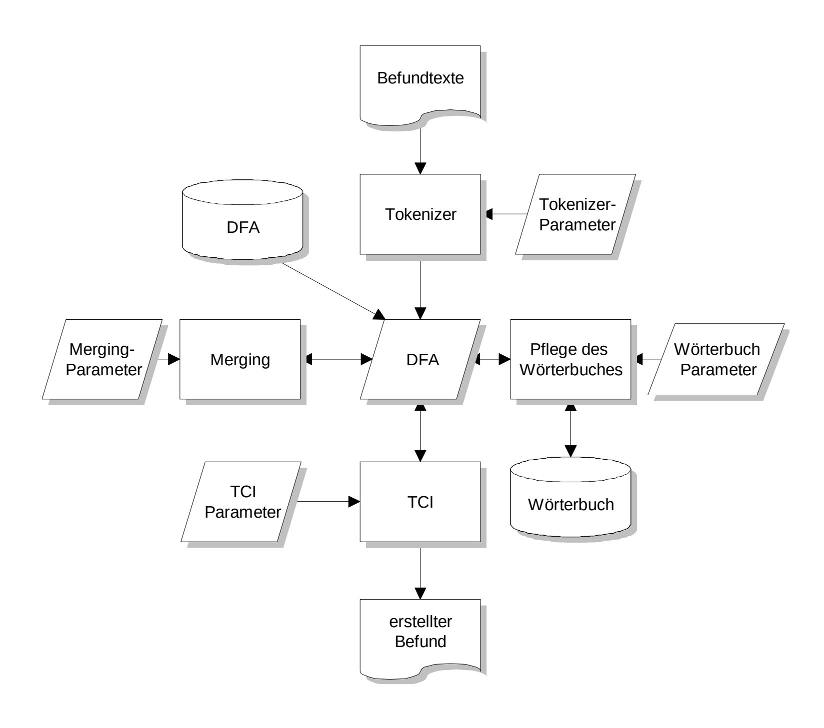


Abbildung 18 Systemarchitektur

Abbildung 18 zeigt eine vereinfachte Darstellung der Systemarchitektur. Am Anfang des Prozesses steht das Einlesen und Vorverarbeiten der Befundungstexte durch einen Tokenizer. Dabei werden die vorverarbeiteten Texte in eine interne Repräsentation, konkret in einen endlichen Automaten (DFA) überführt. Dieser DFA kann um eine zusätzliche sogenannte Meta-Strukturierung erweitert werden, um eine adäquatere

Abbildung der inhaltlichen Gliederung der Befundungstexte zu erreichen und die Ergebnisse des Lernverfahrens zu verbessern.

Der eigentliche Lernvorgang erfolgt nun im Rahmen des sogenannten Mergingprozesses, bei dem äquivalente Zustände des DFA zusammengefaßt werden. Das Ziel dieses maschinellen Lernverfahrens ist es, die den Befundungstexten zugrundeliegende Syntax zu ermitteln. Aus dem so erzeugten DFA, der die syntaktische Struktur der eingelesenen Befunde widerspiegelt, wird nun eine benutzerfreundliche Oberfläche generiert. Diese als TokenComponentInterface bezeichnete Oberfläche ermöglicht es dem Arzt, neue, bisher noch nicht eingegebene Befunde zu erfassen.

Im folgenden sollen die zentralen Schritte vom Einlesen und Vorverarbeiten der Befundungstexte durch Tokenizer und Wörterbuch über den Mergingvorgang, den Aufbau der Benutzeroberfläche bis zur anschließenden Ausgabe des erstellten Befundes skizziert werden. Eine ausführliche Behandlung dieser Schritte erfolgt im zweiten Abschnitt der Diplomarbeit.

6.1 Einlesen und Vorverarbeitung der Befunde

6.1.1 Der Tokenizer

Der Tokenizer verwandelt die eingelesenen Befundungstexte, die aus unterschiedlichen Quellen (Tastatur, Festplatte, Intranet oder Internet) stammen können in eine entsprechende Tokensequenz. Aus dieser Sequenz wird nun eine Datenstruktur aufgebaut, die beginnend mit einem Startzustand und endend mit einem Terminalzustand abwechselnd aus Zuständen und Token besteht.

Des weiteren besteht die Möglichkeit, diese Datenstruktur um eine sogenannte Meta-Struktur zu erweitern, die die inhaltliche Gliederung des Textes widerspiegelt (siehe Kapitel 8.3.1). Dabei kann eine solche Struktur durch Setzen von Markierungen im ursprünglichen Text vorgegeben und im Rahmen der Vorverarbeitung in eine entsprechende Datenstruktur umgesetzt werden.

Um einzelne Worte aus der Token-Sequenz herauszulösen, kann der Benutzer bestimmte Zeichen als Trennzeichen für den Tokenizer definieren. Diese Trennzeichen können beliebig kombiniert werden. Außerdem kann dem System mitgeteilt werden, ob die Trennzeichen selbst als Token interpretiert oder bei der weiteren Verarbeitung unterdrückt werden sollen. Dies kann sinnvoll sein, um die Formatierung des eingelesenen Befundes durch Leerzeilen oder Absatzmarken beizubehalten.

6.1.2 Das Wörterbuch

Das System ermöglicht zum einen das automatische Ersetzen von Wörtern während des Einlesens der Befunde, zum anderen die explizite Veränderung von Wörterbucheinträgen durch den Benutzer. Somit können Synonyme eingefügt,

Abkürzungen expandiert, lange Schreibweisen durch kürzere ersetzt und fehlerhafte Worte automatisch korrigiert werden.

Neben dem einfachen Speichern und Laden ist es möglich, einzelne Wörterbücher aus unterschiedlichen Domänen zusammenzufassen um auf diese Weise ein Wörterbuch zu erstellen, das mehrere Domänen umfaßt. So können im Rahmen der radiologischen Befundung z. B. die Wörterbücher der Domänen CT und MRT der Halswirbelsäule zu einer Domäne Untersuchung der Halswirbelsäule zusammengefaßt werden. Generell ermöglicht der beschriebene Integrationsprozeß eine dynamische Erweiterbarkeit des Domänenwissens.

6.2 Der Mergingvorgang

Der bis zu diesem Zeitpunkt erstellte DFA gibt lediglich eine Repräsentation der bisher eingelesenen Befundungstexte wieder. Dieser Automat erfüllt noch nicht die gewünschte Eigenschaft, weitere bisher noch nicht eingegebene Befunde zu erzeugen. Zu diesem Zweck werden einzelne Zustände des DFA im Rahmen eines maschinellen Lernverfahrens (dem sogenannten Mergingverfahren) zusammengefaßt. Dieser Vorgang führt zu einer Komprimierung des Automaten und somit zu einer Generalisierung der inferierten Sprache. Als Konsequenz dieses Lernvorgangs können nun zusätzliche Befunde auf der Grundlage der bisherigen Befunddaten erstellt werden.

Die umfangreiche Parametrisierbarkeit des Mergingvorganges ermöglicht es dem Benutzer das Lernergebnis zu beeinflussen und somit die zu generierende Benutzeroberfläche seinen Bedürfnissen entsprechend anzupassen.

6.3 Der Aufbau des TokenComponentInterfaces

Das TokenComponentInterface (TCI) ist eine direkte Abbildung des gelernten DFA in eine benutzerfreundliche Oberfläche, die den Arzt durch seinen Befundungsvorgang leitet.

Für jeden Zustand im DFA, der mehr als einen Zustandsübergang hat kann der Arzt zwischen diesen Zustandsübergängen wählen, um in den nachfolgenden Zustand zu gelangen. Die Wahlmöglichkeit an diesen Zuständen wird im TCI durch Radiobuttons repräsentiert. Jeder Radiobutton wird mit der Tokensequenz beschriftet, die den aktuellen Zustand in den Folgezustand überführt. Jeder der ursprünglich eingelesenen Befundungstexte entspricht somit einem bestimmten Pfad im DFA.

6.4 Das Erstellen von Befundungstexten

Der eigentliche Befundungstext setzt sich aus der Folge von Tokensequenzen zusammen, die der Arzt im TokenComponentInterface durch „Anklicken“ der entsprechenden RadioButtons während seines Befundungsvorganges auswählt. Der Befundungsvorgang ist beendet, wenn der Arzt sämtliche Abschnitte (z. B.

Untersuchungstechnik, Fragestellung etc.) behandelt hat und im DFA den Terminalzustand erreicht.

Zusätzlich besteht die Möglichkeit, automatisch einen Briefkopf (enthält z. B. ein Standardanschreiben) vor bzw. einen Brieffuß (enthält z. B. eine Verabschiedungsfloskel) hinter den Befundungstext einzufügen.

Anschließend kann der so erstellte Befund gespeichert oder direkt ausgedruckt werden.

Das bestehende System zeichnet sich durch eine hohe Flexibilität und Adaptivität aus. So kann der Benutzer durch eine Reihe von Parametern sowohl die Eigenschaften des Tokenizers, des Wörterbuches, der Benutzeroberfläche (TCI) als auch des eigentlichen Mergingvorganges verändern und somit das Lernergebnis (die generierte Oberfläche) entsprechend beeinflussen. Eine Zusammenstellung sämtlicher Systemparameter findet sich im Anhang.

7 Verarbeitung der Befundtexte

Damit Texte gelernt werden können, müssen sie zuerst von *mAGENTa* in eine bestimmte Form gebracht werden, damit sie weiter verarbeitet werden können.

Der Text muß dazu in seine einzelnen Bestandteile aufgeteilt werden. Diese Bestandteile (i. A. werden sie als Worte betrachtet) dienen als Grundlage zur Bildung von einzelnen Tokens, die in dem DFA einen Zustand in den nächsten überführen. Diese Aufteilung wird von dem Tokenizer vorgenommen.

Zur Optimierung des Lernergebnisses ist es sinnvoll, für eine Vereinheitlichung der gelernten Texte zu sorgen, indem verschiedene Worte mit der selben Bedeutung auf ein einheitliches Synonym abgebildet werden. Dieses wird durch das Wörterbuch unterstützt.

7.1 Der Tokenizer

Die zu lernenden Texte müssen in einzelne Tokens eingeteilt werden, damit diese gelernt werden können. Der Text wird als Strom von Zeichen eingelesen. Dabei ist es unerheblich, ob dieser Strom von der Tastatur, einem lokalen Speicher, über ein Intranet oder das Internet kommt.

Dieser Strom wird mittels vorher festgelegter Trennzeichen in einzelne Tokens eingeteilt. Ein Token muß dabei nicht zwangsläufig identisch mit einem Wort sein, welches von zwei Leerzeichen begrenzt ist. Es kann eine Reihe von Zeichen angegeben werden, welche als Trennzeichen fungieren sollen.

7.1.1 Die Trennzeichen

Für reale Anwendungen haben die folgenden Trennzeichen sinnvolle Ergebnisse erzielt, da sie eine aus Benutzersicht sinnvolle Einteilung in Tokens boten. Ansonsten würden z.B. Wortendungen nicht richtig erkannt werden.

Trennzeichen	Darstellung
Leerzeichen	" "
Tabulator	" \t "
NeueZeile	" \n "
Wagenrücklauf	" \r "

Trennzeichen	Darstellung
Punkt	" . "
Komma	" , "
Semikolon	" ; "
Bindestrich	" - "

Tabelle 3 Tokenizer Trennzeichen

Diese Trennzeichen können beliebig untereinander kombiniert werden, um unterschiedliche Ergebnisse zu erzielen.

7.1.2 Interpretation der Trennzeichen als eigene Tokens

Es kann *mAGENTa* auch mitgeteilt werden, ob die Trennzeichen selbst als Token interpretiert oder bei der weiteren Verarbeitung unterdrückt werden sollen. Dies kann dann sinnvoll sein, wenn die Formatierung des eingelesenen Befundes durch Leerzeilen oder Absatzendemarken beibehalten werden soll. Würde z.B. ein NeueZeile-Zeichen als Token betrachtet werden, so wird es auch in den DFA aufgenommen, im Graph und folglich im TCI durch "\n" angezeigt.

Ist dies nicht gewünscht, so werden die ausgewählten Zeichen zwar als Trennzeichen der Tokens betrachtet, nicht jedoch in den DFA aufgenommen. Eine etwaige Formatierung durch Steuerzeichen geht hierbei verloren.

7.1.3 Delimiter für MetaZustände

Damit der Lernalgorithmus erkennen kann, wann ein MetaAbschnitt beginnt, und wie dieser Abschnitt betitelt ist, muß vor dem eigentlichen Abschnitt der MetaTitel stehen. Er dient dazu, anstatt eines normalen Zustandes einen MetaZustand zu generieren (siehe Kapitel 7.1 Das MetaStruktur-Konzept).

Der MetaTitel besteht aus mindestens drei Worten: dem Start-Delimiter, der den MetaTitel einleitet, dem eigentlichen MetaTitel und dem End-Delimiter, der den MetaTitel abschließt.

Der MetaTitel muß aus mindestens einem Wort bestehen, kann aber auch ein ganzer Satz sein, der den Inhalt des Abschnittes kurz beschreibt. Die beiden Delimiter stehen neben den übrigen Worte in dem zu lernenden Text.

So führt z.B. der folgende Eintrag in einem eingelesenen Text dazu, daß der MetaZustand *Untersuchungstechnik* gebildet wird, der als Startzustand für den Abschnitt mit dem Thema der Untersuchungstechnik fungiert:

Untersuchungstechnik ## ...

Der Start-Delimiter ist in diesem Fall das Token "#", der End-Delimiter das Token "##". Die drei Punkte signalisiern, daß hier der Text des Abschnittes folgt.

Damit es zu keiner Verwechslung zwischen den Delimitern und den eigentlichen Worten des Textes kommen kann, ist es möglich, beliebige Worte festzulegen, die jeweils als Delimiter fungieren sollen.

7.1.4 Ersetzen von Zahlen

In vielen Texten sind Zahlen zu finden, die nur für einen einzigen Text spezifische Bedeutung haben. So ist z.B. bei EBT-Befunden der Grad der Kalzifikation der Herzkranzgefäße speziell für einen Patienten. Es wird kaum zwei Patienten geben, die über identische Kalzifikationsgrade in den Gefäßen verfügen. Bei mehreren hundert gelernten Befunden würde somit im TCI dem Arzt eine Anzahl von Kalzifikationswerten angezeigt werden, die nicht mehr einfach handhabbar wäre.

Für diese Fälle bietet *mAGENTa* die Option an, alle Tokens, die nur Zahlen als Beschriftung aufweisen, durch jeweils ein festgelegtes Token zu ersetzen. Dieses Token hat die einheitliche Beschriftung "XXX". Durch die Vereinheitlichung der Tokens wird der Lernalgorithmus nun in die Lage versetzt, mehr Zustände zu mergen, sofern es die Mergingregeln erlauben.

Durch die Verringerung der Anzahl der Tokens und Zustände wird nicht nur die Größe des DFA geringer, es verringert sich auch die Anzahl der TokenComponents im TCI. Dies führt zu einer schnelleren Texterstellung, da nicht mehr aus einer großen Menge von TokenComponents ausgewählt werden muß, die mit verschiedenen Zahlen beschriftet sind, sondern es wird nur noch eine TokenComponent mit der Beschriftung "XXX" dargestellt.

Am Ende des Befundungsvorganges hat der Benutzer dann die Möglichkeit, die korrekten Zahlen für den Platzhalter in den erstellten Text einzusetzen.

7.2 Das Wörterbuch

Das Wörterbuch dient zum automatischen Ersetzen von Wörtern.

Es enthält eine Tabelle (Hash-table), in der jedem Wort eine Ersetzung zugeordnet sein kann. Dies dient vor allem dazu, die Befundsprache zu vereinheitlichen. Automatisch können Synonyme ersetzt, Abkürzungen expandiert oder lange Schreibweisen durch kürzere ersetzt werden. Ganz wie es der Benutzer wünscht.

So kann zum Beispiel eingetragen werden, daß das Vorkommen von "Fettsucht" generell durch "Adipositas" ersetzt werden soll. Es wird hiermit die Grundlage für eine einheitliche Nomenklatur gelegt, die von allen Benutzern gleichermaßen genutzt wird.

Da *mAGENTa* von verschiedenen Personen genutzt werden kann, ist es sinnvoll, wenn diese ihre einmal gelernten Texte austauschen und kombinieren, um so ein besseres Gesamtergebnis zu erreichen. Ebenso wie bei DFA ist es auch bei Wörterbüchern möglich, diese anderen nutzbar zu machen, indem ein Benutzer sein erstelltes Wörterbuch abspeichert und es anderen zur Verfügung stellt. Prinzipiell stellt sich hier eine gewisse Unabhängigkeit der Wörterbücher gegenüber den DFA dar.

Die von verschiedenen Benutzern erstellten Wörterbücher können natürlich miteinander kombiniert werden, um so z.B. ein umfassendes und einheitliches Wörterbuch für alle zu erstellen

Das Wörterbuch wird selbständig ohne weitere Aktionen des Benutzers erweitert. Wird ein neuer Text gelernt, ein DFA geladen oder ein Eintrag in eine EmptyComponent des TCI vorgenommen, so werden die Wörter, die bisher noch nicht im Wörterbuch zu finden waren, automatisch eingetragen.

Für jedes Token eines neuen Textes wird in dem Wörterbuch nachgesehen, ob hierfür eine Ersetzung existiert. Falls es ein neues Wort ohne bisherigen Eintrag in das Wörterbuch ist, so wird es mit sich selbst als Ersetzung aufgenommen, andernfalls bekommt das Token die Ersetzung aus dem Wörterbuch als neue Benennung.

Neben dieser automatischen Verfahrensweise beim Einlesen eines Befundes kann das Wörterbuch auch interaktiv genutzt werden, um gezielt einen bestehenden Eintrag durch ein oder mehrere Wörter (1:n-Ersetzung) zu ersetzen. Darüber hinaus ist es auch möglich einen neuen Eintrag für ein oder mehrere Worte und deren Ersetzung (m:n Ersetzung) vorzunehmen. Dies bedeutet, daß Wortgruppen oder ganze Sätze durch andere benutzerdefinierte ersetzt werden können.

Bei den 1:n-Ersetzungen wird ein Token durch eine lineare Folge von Zuständen ersetzt. Dabei wird jeder dieser Zustände jeweils nacheinander durch ein eingegebenes Token in den nachfolgenden Zustand überführt. Bei den m:n-Ersetzungen wird dementsprechend eine Folge von Zuständen durch eine andere ersetzt.

Durch domänenspezifische Wörterbücher wird der Lernvorgang unterstützt und so das Lernergebnis verbessert.

Weitere Informationen zu dem Wörterbuch, insbesondere dessen graphische Gestaltung und Bedienung, ist in dem Benutzerhandbuch in Kapitel 11.5.5 "Das Wörterbuch" angegeben.

8 DFA-Generierung

Dieses Kapitel beschreibt, wie aus den eingelesenen und vorverarbeiteten Befunden (siehe Kapitel 7) zuerst eine formale Repräsentation und anschließend im folgenden Lernschritt aus dieser die zugrundeliegende Syntax der Befundtexte erstellt wird.

Das Ergebnis dieses Lernverfahrens bildet die Grundlage für den nachfolgenden Schritt, der Abbildung des gelernten DFA auf eine benutzerfreundliche Oberfläche. Diese Benutzeroberfläche kann wiederum zur Eingabe neuer Befunde benutzt werden.

Zu Beginn dieses Kapitels werden eine Reihe von Begriffen definiert, die zum Verständnis der nachfolgenden Abschnitte beitragen.

In Kapitel 8.2 werden in allgemeiner Form die grundlegenden Prinzipien der DFA-Generierung im Rahmen der Grammatik-Inferenz beschrieben, um danach konkret auf das verwendete Verfahren einzugehen.

Dabei wird zuerst ein Konzept beschrieben, das die Datenstruktur der endlichen Automaten um eine zusätzliche Strukturierung, die sogenannte Meta-Struktur, erweitert. Diese Meta-Struktur unterteilt den DFA in weitgehend autonome Abschnitte und gibt damit die inhaltliche Gliederung der Befundtexte besser wieder als der ursprüngliche DFA. Der Effekt ist eine Verbesserung der Ergebnisse der nachfolgenden Lernschritte. Im Rahmen des Kapitels 8.3.1 werden diverse Algorithmen auf der Grundlage der Meta-Struktur beschrieben.

Die eigentliche Lernaufgabe, d. h. die Inferenz der Syntax der eingelesenen Befunde findet *innerhalb* einzelner Abschnitte statt. In diesem Zusammenhang werden die Algorithmen von Angluin und Schlimmer/Hermens zur Inferenz k-reversibler Sprachen (einer Teilmenge der regulären Sprachen) aus einer Menge positiver Beispiele (den Befundtexten) vorgestellt.

Zum Abschluß des Kapitels wird das Problem der Übergeneralisierung im Zusammenhang mit der Inferenz k-reversibler Sprachen und deren Auswirkungen auf die Qualität des DFA diskutiert.

8.1 Definitionen

In diesem Abschnitt werden einige grundlegende Definitionen eingeführt, die zum Verständnis dieses Kapitels von Bedeutung sind.

Sei Σ eine endliche, nicht-leere Menge von Symbolen, genannt das *Alphabet*. Dann ist Σ^* die Menge aller Strings über dem Alphabet Σ. Die *Länge* eines Strings a wird mit $|a|$ bezeichnet. Der *leere String* wird mit λ notiert und hat die Länge 0. Die *Konkatenation* zweier Strings a und b wird mit ab bezeichnet.

Falls ein String c=ab gegeben ist, so bezeichnet man a als den *Präfix* und b als den *Suffix* von c.

Eine *Sprache L* ist eine Teilmenge von Σ^*. Die *Menge aller Präfixe einer Sprache L* notiert man mit Pr(L) und definiert sie wie folgt: $Pr(L) = \{a|\ ab \in L\}$. Die *Menge der Suffixe (tails) von L* ist definiert als $T_L=\{b|\ ab \in L\}$. Eine *lexikographische Ordnung* der Strings über dem Alphabet $\Sigma = \{a,b\}$ ordnet alle Strings aus Σ^* zuerst nach ihrer Länge und dann gemäß ihres alphabetischen Wertes. Somit lautet die lexikographische Ordnung über dem Alphabet {a,b}: λ,a,b,aa,ab,ba,bb,aaa,.. . Ein *klassifiziertes Beispiel* ist ein Zweier-Tupel (a,c(a)), wobei $a \in \Sigma^*$ und c eine Klassifizierungsfunktion mit c: $\Sigma^* \rightarrow \{1,0\}$, die folgendermaßen definiert ist: c(a) = 1, falls $a \in L$ und c(a) = 0, falls $a \notin L$. Die *Menge aller positiven Beispiele* S_+ kann mit Hilfe der Klassifikationsfunktion wie folgt definiert werden: $\forall\ a \in S_+$ gilt c(a)=1. Die *Menge aller negativen Beispiele* S_- wird analog definiert: $\forall\ a \in S_-$ gilt c(a)=0.

Ein *deterministischer endlicher Automat (DFA)* ist ein Tupel (Q,δ,Σ,q_0,F), wobei Q die endliche Menge der Zustände, Σ das Alphabet, $q_0 \in Q$ ein ausgezeichneter Startzustand, $F \in Q$ eine endliche Menge von Terminalzuständen (akzeptierenden Zuständen) und δ mit $Q \times \Sigma^* \rightarrow Q$ die Transitionsfunktion beschreibt.

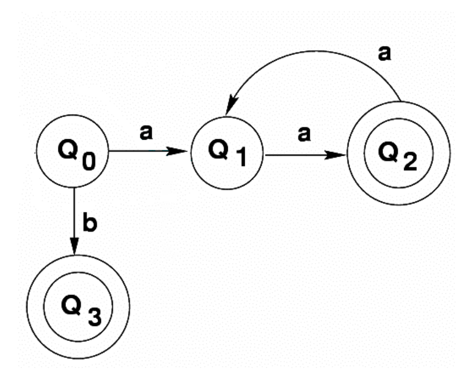


Abbildung 19 Deterministischer endlicher Automat

Abbildung 19 zeigt einen DFA mit $\Sigma=\{a,b\}$, $q_0=Q_0$, $F=\{Q_2, Q_3\}$ (gekennzeichnet durch zwei konzentrische Kreise). Eine konkrete Transition des Automaten wäre z. B. $\delta(Q_1,a) = Q_2$. Eine andere gebräuchliche Bezeichnung für einen DFA ist der Begriff Akzeptor. Dieser Begriff leitet sich aus der Funktion eines DFA ab, für einen beliebigen String w zu entscheiden ob der Automat A den String akzeptiert (formal : $\delta(q_0, w) \in F$).

Die inverse Transitionsfunktion δ^r ist definiert als $\delta^r(q,a) = \{q':q \in \delta(q',a)\}\ \forall a \in \Sigma,\ q \in Q$.

Den inversen Automaten $A^r = (Q, \delta^r,\Sigma,I,F)$ erhält man aus A, indem man die Initial- und Terminalzustände des Automaten vertauscht und die Richtungen sämtlicher Transitionen umkehrt.

Ein String u wird k-Nachfolger (k-follower) eines Zustandes q genannt, genau dann wenn $|u| = k$ und $\delta(q,u)\neq\emptyset$. Analog dazu wird ein String u k-Vorgänger (k-leader) genannt, falls $|u| = k$ und $\delta^r(q,u^r)\neq\emptyset$.

Ein Automat A wird als *kanonisch* bezeichnet bezüglich einer regulären Sprache L(G), falls A eine minimale Anzahl von Zuständen besitzt und L(A)=L(G) gilt.

Eine Menge positiver Beispiele S_+ ist *strukturell vollständig* in Bezug auf einen Automaten A, falls jede Transition von A mindestens einmal benutzt wird und jeder Terminalzustand von A mindestens einen String aus S_+ akzeptiert. Eine strukturell vollständige Menge bezüglich des Automaten in Abbildung 19 wäre $S_+=\{b,aa,aaaa\}$.

Eine *Menge* $S= S_+\cup S_-$ *ist charakteristisch* bezüglich einer regulären Sprache L(G) und eines kanonischen Akzeptors A, falls S strukturell vollständig ist und für jeweils zwei Zustände ein Suffix c existiert, der beide voneinander unterscheidet.

Gegeben sei ein Automat A mit der Menge der Zustände Z. Eine *Partition* π ist eine paarweise disjunkte Vereinigung von Teilmengen von Z.

Sei π eine Partition von S, dann existiert für jedes $s\in S$ genau ein Element von π, das s enthält. Dieses Element wird als *Block* $B(s,\pi)$ bezeichnet.

Gegeben sei ein Automat A und eine Partition π von A. Ein *Quotienten-Automat* A_π entsteht aus A durch Zusammenfassung (Merging) von Zuständen, die zum selben Block von π gehören. Die Partition P_1 des Automaten A in Abbildung 19 lautet $\{\{Q_0\},\{Q_1\},\{Q_2\},\{Q_3\}\}$. Partition P_2 des Automaten A_π in Abbildung 20 lautet $\{Q_0'=\{Q_0,Q_1\},Q_1'=\{Q_2\},Q_2'=\{Q_3\}\}$. Das Ergebnis des Mergingvorgangs ist somit eine Generalisierung der erzeugten Sprache (formal: $L(A) \subseteq L(A_\pi)$).

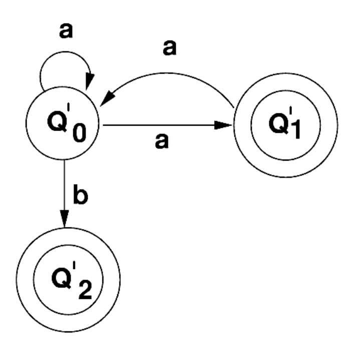


Abbildung 20 Quotientenautomat

Ein Präfixbaum P[7] ist ein Automat, der exakt die Strings aus S_+ akzeptiert. Die formale Definition eines Präfixbaumes $P(S_+) = (Q,\delta,I,F)$ lautet wie folgt : $Q=Pr(S_+)$, $I=(\lambda)$, $F= S_+$ und $\delta(u,a) = ua$ falls u, $a \in Q$. Abbildung 21 zeigt einen Präfixbaum nach der Eingabe der Strings *aa*, *abba* und *baa.*

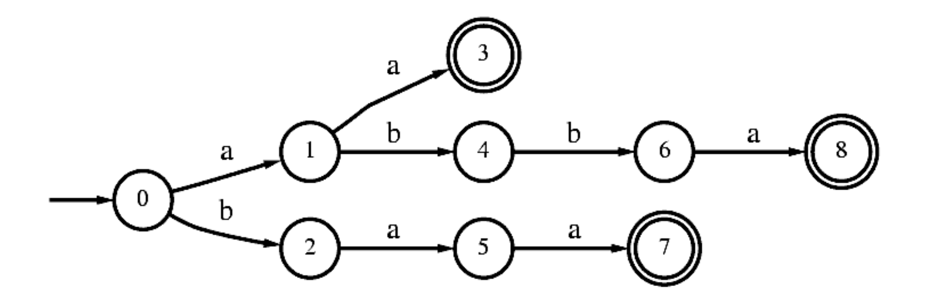


Abbildung 21 Präfixbaum

[7] Im folgenden werden die Begriffe Präfixbaum und Präfixautomat als äquivalent betrachtet

8.2 Das Prinzip der DFA-Generierung

8.2.1 Lernen als Suche

Die Grammatik-Inferenz regulärer Sprachen kann als Suchproblem über der Menge der deterministischen endlichen Automaten aufgefaßt werden. Der Suchraum über der Menge der DFA ist jedoch prinzipiell unendlich. Zur effizienten Inferenz regulärer Sprachen bildet man anfänglich einen Präfixbaum aus der Menge der positiven Beispiele S_+, der ausschließlich die Worte aus S_+ akzeptiert. Der eigentliche Lernprozeß findet im Rahmen der Generalisierung des Präfixbaumes statt. Dieses Kapitel beschäftigt sich nun damit, diesen Prozeß im Allgemeinen und die konkreten Inferenz-Algorithmen im Speziellen zu beschreiben.

8.2.2 Aufbau des Präfixbaumes

Der Aufbau des Präfixbaumes erfolgt inkrementell. Jeder eingelesene Befund wird nach Umwandlung in eine Tokensequenz (siehe Kapitel 7) dem bestehenden Präfixbaum hinzugefügt. Dabei werden die Token der aktuellen Tokensequenz schrittweise mit dem Präfixbaum verglichen. Hierbei wird überprüft, ob für das aktuelle Token der Tokensequenz eine Transition im Präfixbaum existiert. Im positiven Fall wird der Test mit dem nächsten Element der Sequenz wiederholt, bis das Ende der Sequenz erreicht wurde. Im negativen Fall wird für die verbleibenden Token der Sequenz ein neuer Zweig des Präfixbaumes erzeugt.

An Abbildung 21 läßt sich das Prinzip der Generierung eines Präfixbaumes verdeutlichen. Sei *aa* das erste Beispiel, das eingelesen wird. Die Tokensequenz wird in diesem Fall vollständig übernommen. Das zweite eingelesene Beispiel sei *abba*. Das erste Token *a* der Sequenz stimmt noch mit dem Präfixbaum überein, für das zweite Token *b* existiert hingegen keine Transition im Präfixbaum. Es wird also ein neuer Zweig mit der Sequenz *bba* angefügt. Beim dritten eingelesenen Beispiel stimmt bereits das erste Token *b* nicht mit dem Präfixbaum überein. Der neue Zweig *bba* wird direkt an den Startzustand des Baumes angehängt.

8.2.3 Generalisierung des Automaten

Der Verband (*Lattice)* der endlichen Automaten ist definiert als die Menge aller Partitionen des Präfix-Automaten. Die einzelnen Elemente des Verbandes sind Quotienten-Automaten im Sinne der Definition und entstehen aus dem Präfix-Automaten durch Merging von Zuständen, die zum gleichen Block einer Partition gehören. Ein Quotienten-Automat π_1 ist allgemeiner als ein Quotienten-Automaten π_2, falls π_1 durch eine Zusammenfassung von Blöcken von π_2 entstanden ist. Aufgrund dieser Relation läßt sich ein partiell geordneter Suchraum erzeugen. Dieser Suchraum besteht aus der Potenzmenge der Zustände des Präfix-Automaten und ist somit exponentiell in der Anzahl der Zustände.

Wie bereits in Kapitel 5 diskutiert, setzt an dieser Stelle der im System verwendete Algorithmus an. Der Ansatz ist dabei, die Klasse der zu lernenden deterministischen Automaten einzuschränken. Dabei wird ein auf den Arbeiten von [Angluin 1983] und [Schlimmer et al. 1993] basierender Algorithmus zur Inferenz von k-reversiblen Sprachen verwendet, der es ermöglicht, den eingeschränkten Suchraum in polynomieller Zeit zu bearbeiten.

Die Forderung, daß sich der gesuchte Automat mit minimaler Anzahl von Zuständen im betrachteten Hypothesenraum befindet kann garantiert werden, wenn die vorliegende Menge positiver Beispiele S_+ strukturell vollständig ist (siehe hierzu auch [Dupont et al. 1994]). Diese Eigenschaft ist im Falle des Präfix-Automaten gegeben.

Der Präfix-Automat eignet sich im Besonderen als Ausgangspunkt für die Erzeugung des Verbandes, da er die speziellste Hypothese ist, die konsistent ist mit der gegebenen Beispielmenge.

8.3 Der Merging-Algorithmus

In Tabelle 4 wird der zentrale Algorithmus der adaptiven Komponente beschrieben.

```
Eingabe :  Ein DFA A  mit einer Menge von Zuständen Z,
           einer Menge von Meta-Zuständen M mit M∩Z ≠0 und
           einer Regelmenge R
Ausgabe :  Ein DFA A'

Für alle Meta-Zustände m ∈M
   Ermittle die Menge aller von m erreichbaren Zustände EZ ⊆ Z
   ERZEUGE_PRÄFIXBAUM(EZ)
   Ermittle die Menge aller Zustände des Präfixbaumes PZ ⊆ EZ
   Wiederhole
      Für alle Zustandspaare ZP (z1,z2) mit z1,z2 ∈ PZ
         Falls eine der ausgewählten Regeln r ∈ R anwendbar ist
         MERGE (z1,z2)
   Bis kein Zustandspaar mehr die Merging-Bedingungen erfüllt
```

Tabelle 4 Skizze des Merging-Algorithmus

Ausgangspunkt für den eigentlichen Mergingvorgang ist der bereits erwähnte Präfixbaum, genauer die Zustandsmenge des Präfixbaumes. Dieser Präfixbaum repräsentiert ausschließlich die Menge der positiven Beispiele, also konkret die eingelesenen Befunde. Der Aufbau des Präfixbaumes wurde bereits in Kapitel 8.2.2 erläutert.

Der nachfolgend beschriebene Mergingprozess dient nun der Generalisierung des Präfix-Automaten.

Das Testverfahren durchläuft den Automaten rückwärts, beginnend mit den Terminalzuständen, und numeriert die Zustände gemäß eines BFS-Durchlaufes auf dem inversen Automaten. Entsprechend dieser Numerierung werden nun die Zustandspaare gebildet, für die getestet wird ob die Voraussetzungen für eine Anwendung der ausgewählten Regeln erfüllt sind. Dieser Vorgang wird solange wiederholt bis kein Zustandspaar mehr die Merging-Bedingungen erfüllt.

Abbildung 22 visualisiert den Mergingvorgang zweier Zustände. Im Rahmen des Mergingvorganges wird ein Zustand ausgewählt, der aus dem DFA gelöscht werden soll. Bei der Auswahl des zu löschenden Zustandes muß beachtet werden, daß, falls mindestens einer der Zustände ein Terminalzustand ist, der verbleibende Zustand ebenfalls terminal ist. Somit wird gewährleistet, daß immer mindestens ein Terminalzustand existiert. In der Abbildung wurde Q_2 als zu löschender Zustand ausgewählt. Das bedeutet, daß das Nachfolger-Token von Zustand Q_2 „d“ vor den verbleibenden Zustand Q_1 gehängt wird. Die Vorgänger-Token „a“, “b“ und „c“ werden ebenfalls dem verbleibenden Zustand Q_1 zugeordnet. Auf der rechten Seite der Abbildung 22 ist der verbleibende Zustand $Q_{1,2}$ nach Anwendung der beschriebenen Operationen zu sehen.

Den Token im DFA sind neben ihrem Label noch die Häufigkeit ihres Auftretens (ihre Frequenz) in den Befundtexten zugeordnet. Faßt man nun zwei Zustände inklusive ihrer Vorgänger- und Nachfolger-Token zusammen müssen die Frequenzwerte der Token des zu löschenden Zustandes denen des verbleibenden aufsummiert werden. Die Frequenzwerte spielen eine wichtige Rolle bei der Generierung der Benutzeroberfläche.

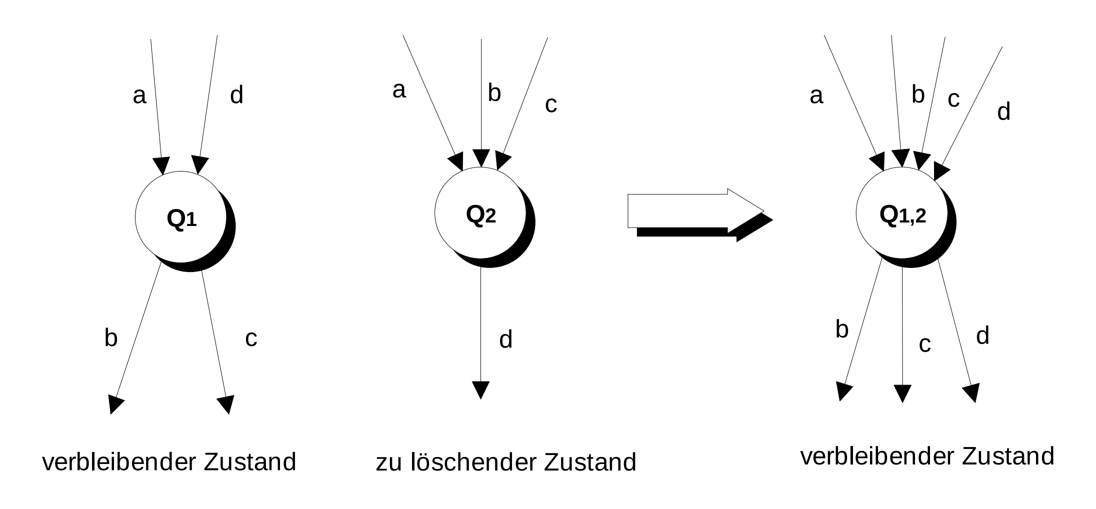


Abbildung 22 Mergingvorgang

8.3.1 Die Meta-Struktur

Texte die eine bestimmte inhaltliche Gliederung in Form von festgelegten Abschnitten aufweisen eignen sich im besonderen für die Verwendung im vorgestellten System. Diese Struktur findet sich u. a. in medizinischen Befunddokumentationen (insbesondere radiologischer Art) und legt den Ansatz nahe, die Repräsentationsform der Befunddaten im System, die deterministischen endlichen Automaten, um eine Meta-Struktur, die die Abschnittsstruktur der Befunde widerspiegelt zu erweitern.[8]

Eine Voraussetzung, die das System an die Eingabedaten stellt, ist, daß zu Beginn des Lernvorgangs bereits die Menge der Abschnitte bekannt ist, die in den Befundtexten verwendet wird. Da in der Praxis der radiologischen Befundung wie auch innerhalb der meisten anderen Domänen die Struktur eines Befundes standardisiert ist, stellt diese Vorgabe jedoch keine wesentliche Einschränkung dar.

8.3.1.1 Motivation

Die wichtigsten Argumente für die Erweiterung der Datenstruktur um eine Meta-Struktur sind die Einschränkung des Hypothesenraumes, eine Verringerung der Rechenzeit und eine Verminderung von unerwünschten Lernergebnissen.

8.3.1.1.1 Einschränkung des Hypothesenraumes

Die Strukturierung endlicher Automaten bedeutet eine erhebliche Eingrenzung des Hypothesenraumes und damit eine Beschränkung der möglichen Lernergebnisse des Systems (*language bias*). Sei n die Anzahl der Strings eines Befundes und somit die Anzahl der Token eines unstrukturierten Automaten A_1 (d. h. eines Automaten ohne explizite Meta-Struktur), m die Anzahl der Abschnitte im strukturierten Automaten A_2 und w die Anzahl der Worte bzw. die Kardinalität des Eingabealphabets der Automaten. Die Zahl der Hypothesen beträgt im Fall des unstrukturierten Automaten w^n. Unter der vereinfachenden Annahme, daß die Zustandsmengen der einzelnen Abschnitte in einem strukturierten Automaten gleiche Kardinalität besitzen ergibt sich für die Zahl der Hypothesen in diesem Fall $m \cdot w^{\frac{n}{m}}$. Sei n = 100, m = 10 und w = 80 (realistische Werte eines durchschnittlichen radiologischen Befundes) so ergibt sich im unstrukturierten Fall ein Suchraum von $80^{100} \cong 10^{190}$ Hypothesen, im strukturierten Fall beträgt die Größe des Suchraumes $10 \cdot 80^{\frac{100}{10}} = 10 \cdot 80^{10} \cong 10^{20}$ Hypothesen.

---

[8] Für eine ausführlichere Diskussion des Komplexes der radiologischen Befundung insbesondere des Befundungsvorgangs, des Aufbaus eines radiologischen Befundes und der radiologischen Fachsprache sei an dieser Stelle auf das Kapitel 3 verwiesen.

8.3.1.1.2 Verringerung der Rechenzeit

Der zugrundeliegende Algorithmus hat unabhängig von der gewählten Regelmenge R eine Laufzeit von $O(n^3)$, wobei n die Anzahl der Zustände des Automaten ist. Diese Anzahl steht in direktem Zusammenhang mit der Länge der Beispiele. Sei nun A_S ein strukturierter Automat mit m Meta-Abschnitten (Zustandsmengen) $a_1,..,a_m$ und der Größe der zugehörigen Zustandsmengen $n_1,..,n_m$ so ergibt sich für die Abschätzung der Rechenzeit für einen Mergingvorgang auf A_S die Formel $n_1^3+..+n_m^3$. Zum besseren Vergleich der Laufzeiten sei nun eine Gleichverteilung der Zustandsmengen über die m Abschnitte angenommen. Damit folgt für die resultierende Laufzeit: $m \cdot \left(\frac{n}{m}\right)^3 = \frac{n^3}{m^2}$.

Dieses Ergebnis zeigt, daß eine Strukturierung in kleinere Abschnitte neben der Einschränkung des Hypothesenraumes eine signifikante Verringerung der Rechenzeit des Algorithmus bedeutet.

8.3.1.1.3 Einschränkung von unerwünschten Lernergebnissen

Die Meta-Struktur stellt eine explizite Nebenbedingung des Lernverfahrens in Form einer Gliederung des DFA dar. Diese Struktur dient der Einschränkung unerwünschter Lernergebnisse, da der Merging-Algorithmus ausschließlich innerhalb einzelner Abschnitte angewendet wird. Dieses Verfahren verhindert, daß z. B. Zustände aus dem Abschnitt *Untersuchungstechnik* mit Zuständen aus dem Abschnitt *Beurteilung* verglichen und fälschlicherweise zusammengefaßt werden.

8.3.2 Merging von Meta-Zuständen

8.3.2.1 Metaregeln

Der Mergingvorgang in Bezug auf die Meta-Struktur des DFA unterscheidet sich deutlich vom Mergingvorgang innerhalb der Meta-Abschnitte. Im ersten Fall ist das Ziel, die vorgegebene äußere Struktur der Befunde hinsichtlich der Abfolge einzelner Abschnitte zu beschreiben, im zweiten Fall hingegen findet der eigentliche Lernvorgang auf der syntaktischen Ebene der Befundinhalte einzelner Abschnitte statt. Einen anderen Schwerpunkt setzte im Gegensatz dazu H. Ahonen, die sich in ihrer Dissertation [Ahonen 1996] vor allem auf den *Aufbau* strukturierter Texte (z. B. Wörterbücher, Enzyklopädien etc.) konzentrierte und dabei von den Inhalten einzelner Abschnitte abstrahierte.

Zum besseren Verständnis der folgenden Abbildungen sei darauf hingewiesen, daß ein Kreis jeweils einen Meta-Zustand darstellt. Dieser Zustand hat die Eigenschaft, Start-Zustand eines Meta-Abschnittes zu sein und diesen im DFA zu repräsentieren. Jedem Meta-Zustand ist ein entsprechender Bezeichner (Meta-Titel) zugeordnet. Ein Rechteck ist als Abstraktion für die Menge aller von M erreichbaren und somit für diesen Abschnitt relevanten Zustände im Rahmen des Lernverfahrens zu verstehen. Die

Bezeichnung der Abschnitte (z. B. U_2) ist dabei lediglich ein Hilfsmittel zur Vereinfachung der Darstellung.

Im folgenden werden die beiden Mergingverfahren auf der Ebene der Meta-Struktur beschrieben. Die beiden Regeln berücksichtigen dabei in jeweils unterschiedlichem Grad die Meta-Struktur der verwendeten Befunde.

Den Ausgangspunkt des Mergingvorganges bildet dabei der erste eingelesene Befund, der bereits die vollständige Menge der Abschnitte enthält.

8.3.2.2 Präskriptive Meta-Regel

Dem präskriptiven Ansatz liegt die Idee zugrunde, daß mit der Meta-Struktur des ersten eingelesenen Befundes eine optimale Struktur vorliegt, die während des gesamten Lernvorganges hinweg beibehalten werden soll.

Abbildung 23 soll die Funktionsweise des Algorithmus verdeutlichen. Befund 1 sei der erste Befund, der eingelesen wurde und seine Meta-Struktur somit Grundlage für alle nachfolgenden Befunde. Vereinfacht gesagt werden die Meta-Abschnitte des *neuen* Befund-Graphen (Befund 2) extrahiert und an der entsprechende Position des *alten* Befund-Graphen (Befund 1) eingefügt. Beispielsweise wird der Abschnitt F_2 des neuen Befundes mit dem zugehörigen Meta-Titel *Fragestellung* zwischen dem Meta-Zustand des alten Befundes mit gleichem Meta-Titel und dessen direkten Nachfolger im Meta-Graphen eingeordnet. Betrachtet man das Ergebnis des präskriptiven Merging-Vorganges so wird deutlich, daß die Meta-Struktur des neuen Befundes, insbesondere die Abfolge der Abschnitte, nicht berücksichtigt wurde. Dies erfolgt unter der Prämisse, daß nur die Inhalte der Abschnitte der neuen Befunde von Bedeutung sind, nicht jedoch ihre Meta-Struktur.

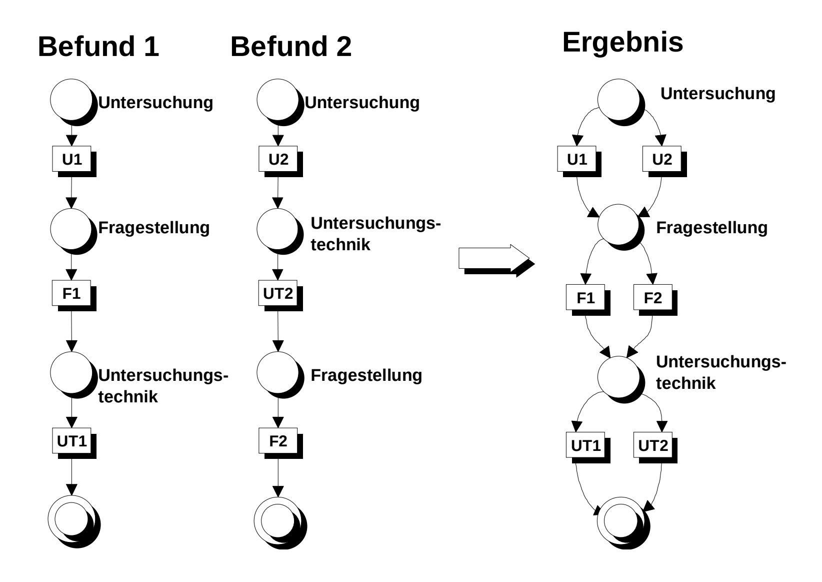


Abbildung 23 Präskriptive Metaregel

8.3.2.3 Deskriptive Meta-Regel

Im Falle des deskriptiven Ansatzes ist das Ziel nicht wie beim präskriptiven Ansatz, eine einmal festgelegte Struktur beizubehalten, sondern eine Struktur zu erstellen die den Aufbau aller Befunde, insbesondere die Reihenfolge der Abschnitte innerhalb einzelner Befunde berücksichtigt.

Abbildung 24 verdeutlicht diesen Ansatz. In Befund 1 weisen die Abschnitte die Reihenfolge *Untersuchung*, *Fragestellung* und *Untersuchungstechnik* auf. In Befund 2 liegt folgende Abfolge der Abschnitte vor: *Untersuchung, Untersuchungstechnik* und *Fragestellung.* Im Falle des ersten Befundes folgt z. B. nach dem Abschnitt *Untersuchung* der Abschnitt *Fragestellung*. Im zweiten Fall folgt nach *Untersuchung* die *Untersuchungstechnik.* Das Ergebnis zeigt, daß die Reihenfolge der Abschnitte in Befund 2 berücksichtigt wurden.

Die Wahl zwischen präskriptivem und deskriptivem Ansatz hängt somit vor allem davon ab, ob man der Reihenfolge der Abschnitte Bedeutung beimißt oder nicht.

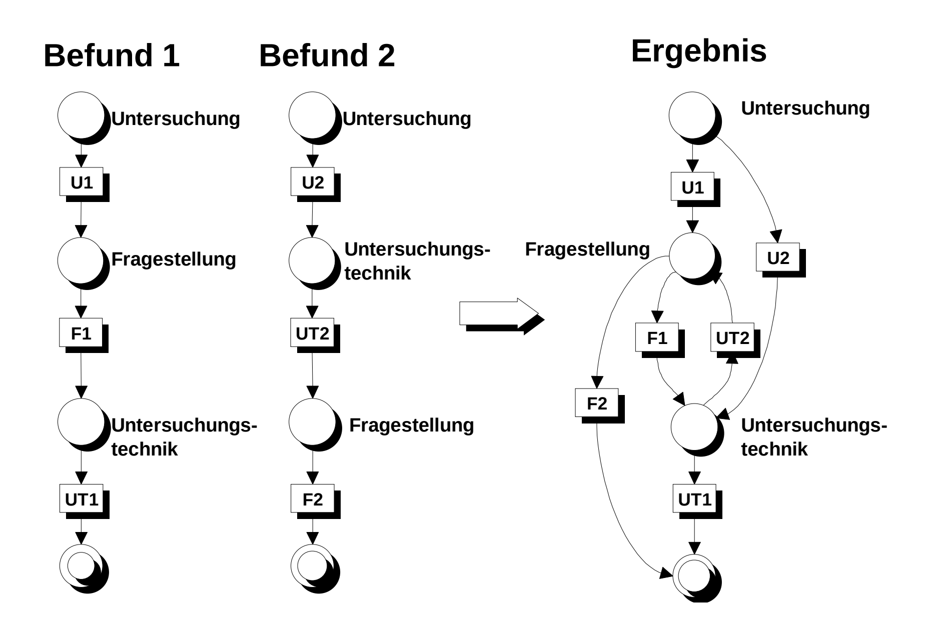


Abbildung 24 Deskriptive Metaregel

Abbildung 25 zeigt den Meta-Graphen nach Verarbeitung eines Befundes mit den Meta-Abschnitten Untersuchung, Fragestellung, Untersuchungstechnik, Beurteilung, Segment L1/2, Segment L2/3, Segment L3/4, Segment L4/5, Segment L5/S1, Nebenbefund, Muskulatur sowie eines zweiten Befundes mit den Abschnitten Untersuchung, Fragestellung, Untersuchungstechnik, Beurteilung, Segment L3/4, Segment L5/S1, Nebenbefund, Muskulatur. Im zweiten Befund werden die Abschnitte Segment L1/2, Segment L2/3 sowie Segment L4/5 nicht behandelt. In der Abbildung spiegelt sich dies in der dargestellten Struktur des Meta-Graphen wider.

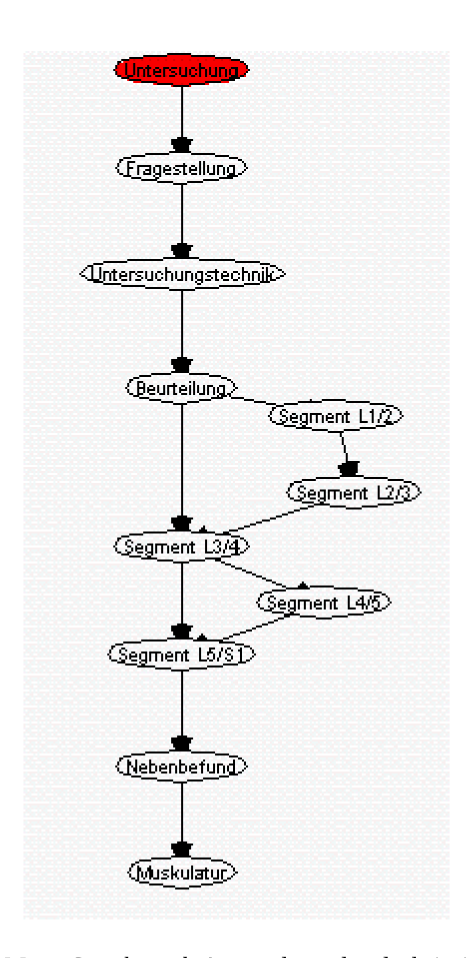


Abbildung 25 Meta-Graph nach Anwendung der deskriptiven Metaregel

Der deskriptive Ansatz findet also immer dann Verwendung, wenn nicht nur die Struktur innerhalb der Meta-Abschnitte sondern auch die Meta-Struktur der Befunde, insbesondere die Abfolge der Abschnitte gelernt werden soll.

8.3.2.4 Verarbeitung unvollständiger Befunde

Eine Option des Systems ist die Verarbeitung unvollständiger Befunde. Eine bereits in Kapitel 8.3.1 beschriebene Voraussetzung ist hierbei, daß der erste Befund bereits die Menge aller in den Befunden vorkommenden Meta-Abschnitte enthält. Der folgenden Abbildung 26 liegt der präskriptive Ansatz zugrunde, d. h. die in Befund 1 vorgegebene Meta-Struktur bleibt auch nach Einlesen des neuen Befundes 2 erhalten. Der Unterschied zu Abbildung 23 liegt darin, daß es erlaubt ist, bei neuen Befunden lediglich eine Teilmenge der vollständigen Abschnittsmenge anzugeben.

Dies hat den Vorteil, daß es bei der Eingabe der Befundtexte ausreicht, sich auf pathologische Beschreibungen zu beschränken. Nicht aufgeführte Abschnitte wie z. B. *Segment 1/2, Segment 2/3* und *Segment 4/5* in Abbildung 25, werden dabei als unauffällig bzw. ohne Befund gewertet ohne dies explizit angeben zu müssen.

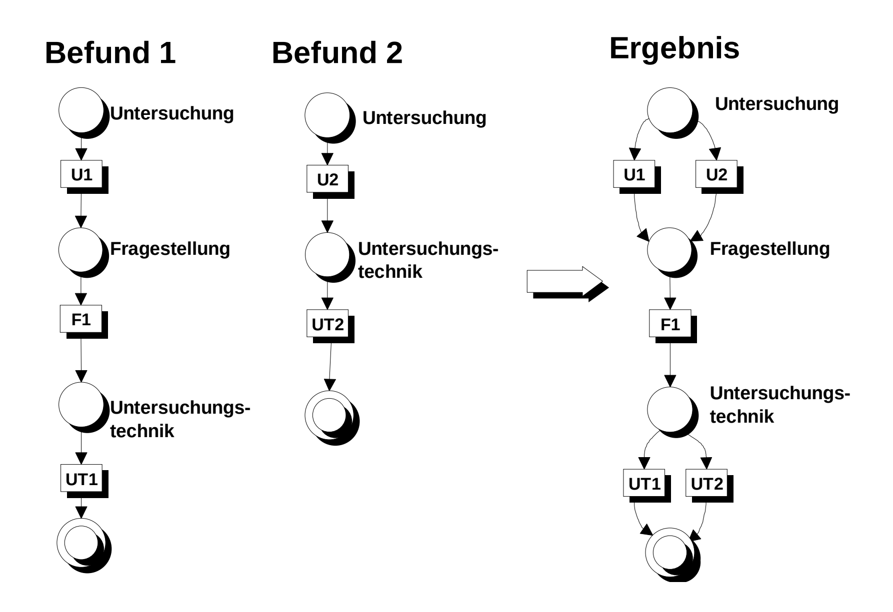


Abbildung 26 Verarbeitung unvollständiger Befunde

8.3.2.5 Verarbeitung von Befunden mit mehrfach auftretenden Meta-Titeln

Neben der soeben beschriebenen Möglichkeit, unvollständige Befunde einzugeben, können auch Befunde verarbeitet werden, in denen einzelne Abschnitte mehr als einmal aufgeführt werden. Diese Option ermöglicht es Befunde zu verarbeiten, bei denen die Inhalte einzelner Abschnitte beliebig innerhalb des Befundtextes verteilt sind. Wie in Abbildung 27 zu erkennen ist der Abschnitt *Untersuchung* in Befund 2 in zwei Teile U_2 und U_3 aufgeteilt. Das Ergebnis des Mergingvorgangs zeigt, daß U_2 und U_3 zusammengefaßt worden sind.

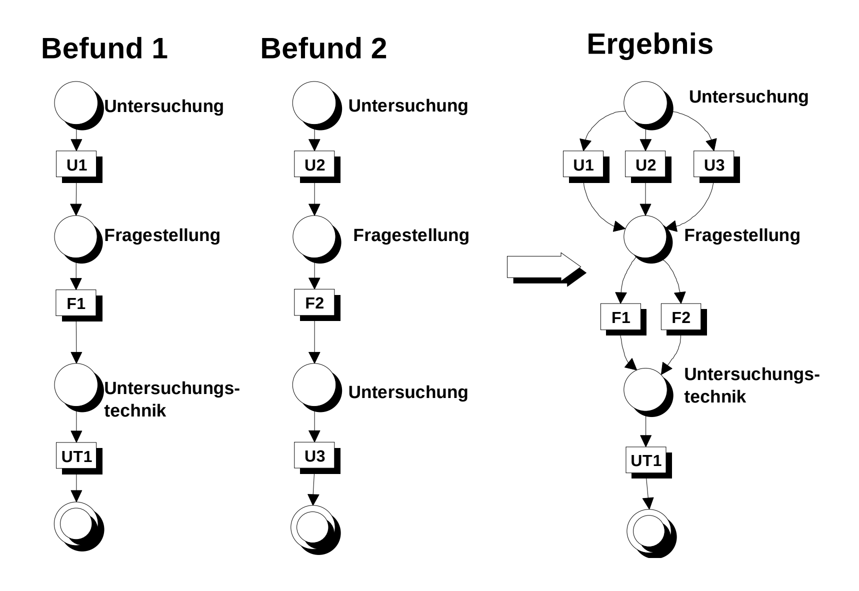


Abbildung 27 Verarbeitung von Befunden mit identischen Meta-Titeln

8.3.3 Merging von Zuständen

Wie bereits in Kapitel 5 erwähnt basieren die verwendeten Verfahren zum Merging von Zuständen auf den Algorithmen von Angluin und Schlimmer/Hermens zur Inferenz k-reversibler Sprachen. Im folgenden Abschnitt werden aus diesem Grund sowohl die Klasse der k-reversiblen Sprachen als auch die genannten Algorithmen mit besonderem Schwerpunkt auf der Beschreibung der jeweiligen Regelmengen intensiver untersucht.

8.3.3.1 Die Klasse der k-reversiblen Sprachen

[Berwick et al. 1987] definiert die Klasse der k-reversiblen Sprachen wie folgt: „A regular language is k-reversible, where k is a non-negative integer, if whenever two prefixes whose last k words match have a tail in common, then the two prefixes have all tails in common. In other words a deterministic finite state automaton is k-reversible if it is deterministic with lookahead k when its sets of initial and final states are swapped and all of its arcs are reversed.“

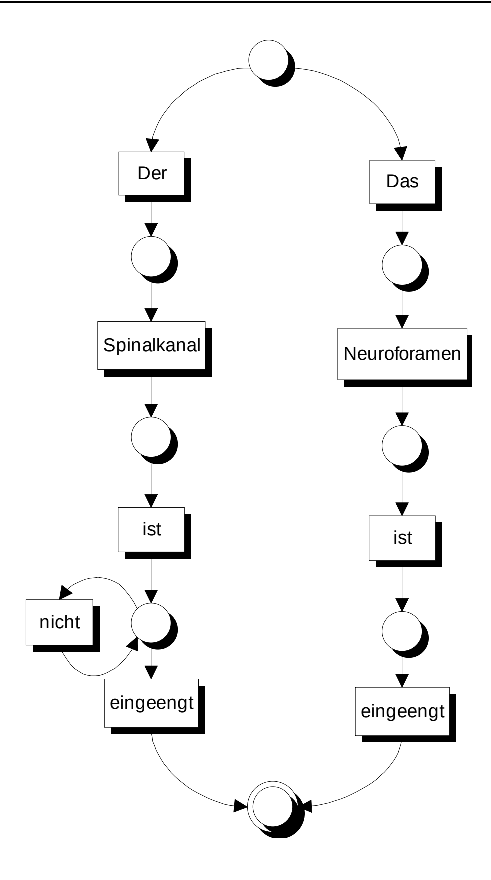


Abbildung 28 Nicht k-reversibler Automat

Anhand von Abbildung 28 kann das Kriterium der k-Reversibilität veranschaulicht werden. Einerseits haben die Präfixe „*Der Spinalkanal*“ und „*Das Neuroforamen*“ zwar den gemeinsamen Suffix „*ist eingeengt*“ jedoch kann der Satz „*Das Neuroforamen ist nicht eingeengt*“ nicht erzeugt werden. Die Präfixe teilen in diesem Fall also nicht alle Suffixe. Andererseits ist leicht ersichtlich, daß nach Vertauschung von Terminal- und Startzustand und nach Umkehrung aller Kanten der DFA nicht mehr deterministisch ist.

8.3.3.1.1 Der Algorithmus von Angluin

Der Algorithmus von Angluin ist ein Verfahren zur Grammatik-Inferenz und erzeugt aus einer Menge von positiven Beispielen (Befundtexten) einen k-reversiblen Automaten. Dabei wird für alle Zustandspaare des ursprünglichen Automaten überprüft ob eine der in Tabelle 5 angegebenen Regeln erfüllt ist. In diesem Fall werden die Zustände des entsprechenden Zustandspaares zusammengefaßt (siehe Kapitel 1.3). Dieser Vorgang wird solange wiederholt bis keine der aufgeführten Regeln mehr erfüllt ist.

```
Another state arcs to both states on the same word OR
Both states have a common k-Leader AND either
   (a) both states are accepting states, OR
   (b) both states arc to a common state via the same word.
```

Tabelle 5 Mergingregeln von Angluin

Im folgenden werden die Regeln von Angluin zur Inferenz k-reversibler Sprachen, die Erweiterungen von Schlimmer und Hermens und eine Modifikation einer Regel von Schlimmer und Hermens detailliert beschrieben. Da das Lernergebnis entscheidend von der Kombination der o. g. Regeln beeinflußt wird ist ein Verständnis der Regeln im Einzelnen die Basis für eine Verbesserung des Lernergebnisses und somit für eine erhöhte Benutzerfreundlichkeit der erzeugten Oberfläche.

8.3.3.1.1.1 Regel 1

Die erste Regel garantiert den Determinismus des erzeugten Automaten. Das zu betrachtende Zustandspaar in Abbildung 29 ist (S_1,S_2). Falls diese Zustände einen direkten gemeinsamen Vorgängerzustand S_x besitzen und sowohl $\delta(S_x,A)=S_1$ als auch $\delta(S_x,A)=S_2$ gilt, also ein nichtdeterministischer Automat vorliegt, werden die Zustände S_1 und S_2 zur Wiederherstellung des Determinismus zu einem neuen Zustand $S_m \in \{S_1, S_2\}$zusammengefaßt. Dabei werden die Vorgänger- und Nachfolgerlisten der beiden Zustände gemäß dem in Abbildung 22 beschriebenen Mergingvorgang angepaßt und der Frequenzwert des verbleibenden Tokens A wird dabei um den Wert des gelöschten Tokens erhöht.

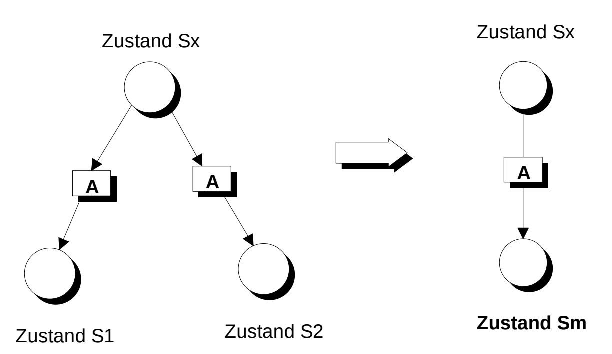


Abbildung 29 Mergingregel 1

8.3.3.1.1.2 Regel 2a

Diese Regel faßt alle Terminalzustände zusammen, die einen gemeinsamen k-Vorgänger besitzen. Im Spezialfall k = 0 führt die Anwendung dieser Regel dazu, daß der DFA nur noch einen Terminalzustand enthält. Analog zur Regel 1 werden auch hier die Vorgänger- und Nachfolgerlisten der beiden Zustände entsprechend angepaßt.

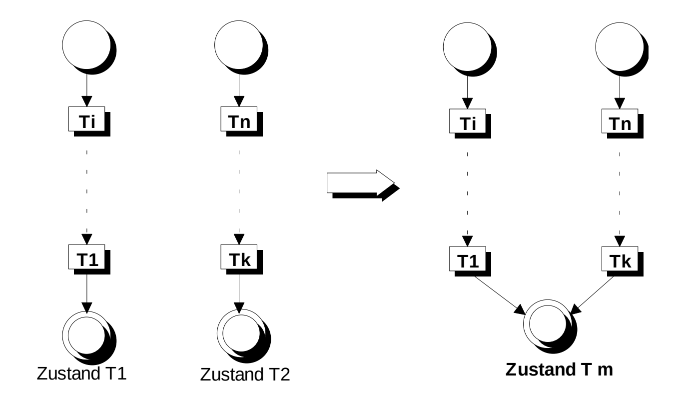


Abbildung 30 Mergingregel 2a

8.3.3.1.1.3 Regel 2b

Praktisch werden die Zustände S_x und S_y zusammengefaßt, wenn zum einen $T_1=T_k,..,T_i=T_n$ gilt, also S_x und S_y identische k-Vorgänger besitzen, und zum anderen die beiden Zustände S_x und S_y jeweils über Token mit identischem Label (hier A) in den gemeinsamen Nachfolgezustand S_1 übergehen. Die Liste der Vorgänger- bzw. Nachfolger-Token des verbleibenden Zustandes wird um die des zu löschenden Zustandes erweitert und der Frequenzwert des verbleibenden Tokens um den des zu löschenden erhöht.

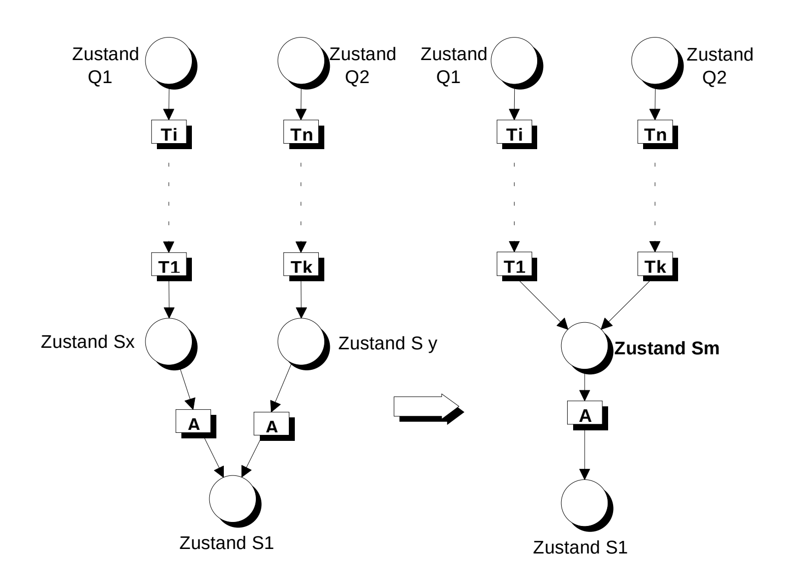


Abbildung 31 Mergingregel 2b

8.3.3.1.1.4 Eigenschaften des Algorithmus

Wie bereits in Kapitel 5 erwähnt sind die Klasse der regulären Sprachen sowie deren äquivalenter Repräsentationsformalismus, die deterministischen endlichen Automaten, bekannt und ihre formalen Eigenschaften gut untersucht. Diese Eigenschaften gelten insbesondere für die von Angluin beschriebene Klasse der k-reversiblen Sprachen, einer Teilmenge der regulären Sprachen.

Man kann unter der Voraussetzung der k-Reversibilität einer Sprache L z. B. die Identifikation im Grenzwert und die Reihenfolge-Invarianz für eine gegebene Menge positiver Beispiele, konkret der zur Verfügung stehenden Befundmenge, garantieren.

Außerdem stehen im Fall der radiologischen Befundung keine zusätzlichen Informationen, insbesondere in Form negativer Beispiele zur Verfügung. Da der Algorithmus von Angluin ausschließlich aus positiven Beispielen lernt kann er als Basis für entsprechende Inferenz-Algorithmen im Rahmen des Systems verwendet werden.

Umfangreiche Evaluierungen des Systems anhand unterschiedlicher radiologischer Domänen haben gezeigt, daß die auf der Grundlage des Algorithmus von Angluin erzeugten Grammatiken eine ausreichende Ausdrucksstärke in Bezug auf die radiologische Befundungssprache besitzen.

Des weiteren wurde festgestellt, daß die auf der Basis der verwendeten Befunddaten erzeugten Automaten 2-reversibel sind.

Ein weiterer Vorteil des Algorithmus in Bezug auf den praktischen Einsatz ist die niedrige polynomielle Laufzeit. Die Laufzeit von $O(kn^3)$, wobei n die Anzahl der Zustände ist ergibt sich im Übrigen dadurch, daß jedes der n^2 Zustandspaare im Worst-Case n-mal betrachtet wird. Der Faktor k läßt sich dadurch erklären, daß jeweils k Vorgänger-Token der betrachteten Zustandspaare untersucht werden müssen.

8.3.3.1.2 Die Erweiterungen von Schlimmer und Hermens

[Schlimmer et al. 1993] verwendeten in ihrer Arbeit den bereits beschriebenen Algorithmus von Angluin zur Inferenz k-reversibler Sprachen und ergänzten die Regelmenge[9] um zwei weitere Regeln (siehe Tabelle 6).

```
1. Another state arcs to both states on the same word OR
2. Both states have a common 0-Leader AND either
     (a) both states are accepting states OR
     (b) both states transition to a common state via the same word.
3. Both states have a common 1-Leader AND either
     (a) both states transition to a common state via any word OR
     (b) one transitions to the other via any word.
```

Tabelle 6 Mergingregeln von Schlimmer und Hermens

Die Regeln 1, 2a und 2b entsprechen der Regelmenge von Angluin für einen 0-reversiblen Automaten (siehe Tabelle 5). Die Regeln 3a und 3b wurden zusätzlich von Schlimmer und Hermens eingeführt um die Qualität der generierten Benutzeroberfläche zu verbessern. Beiden zusätzlichen Regeln ist gemein, daß sie den Generalisierungs-

[9] An dieser Stelle sei ausdrücklich darauf hingewiesen, daß im vorliegenden System die Wahl des k-Vorgängerwertes bezüglich der in Tabelle 6 aufgeführten Regeln unbeschränkt ist.

vorgang beschleunigen und im Vergleich zur ursprünglichen Regelmenge von Angluin zum Lernen der Syntax deutlich weniger Befunde benötigen.

Abbildung 32 macht in diesem Zusammenhang eine Schwäche des Algorithmus von Angluin deutlich. Viele der Zustände in dieser Abbildung könnten sinnvollerweise zusammengefaßt werden. Allein auf der Grundlage der Angluin-Regeln hätte es jedoch noch vieler Beispiele bedurft, um den DFA in Abbildung 33 zu generieren. In Abbildung 32 tritt häufig die Situation auf, daß viele Tokensequenzen, bei denen dem String *Size* eine Größenangabe folgt, im selben Zustand enden. Die Angluin-Regel 2b findet in diesen Fällen jedoch keine Anwendung, da sich die Größenangaben voneinander unterscheiden. Die Voraussetzungen zur Anwendung der Regel 3a von Schlimmer und Hermens sind hingegen in diesen Fällen erfüllt. Abbildung 33 zeigt das Ergebnis des Generalisierungsprozesses auf der Grundlage der Regeln von Schlimmer und Hermens.

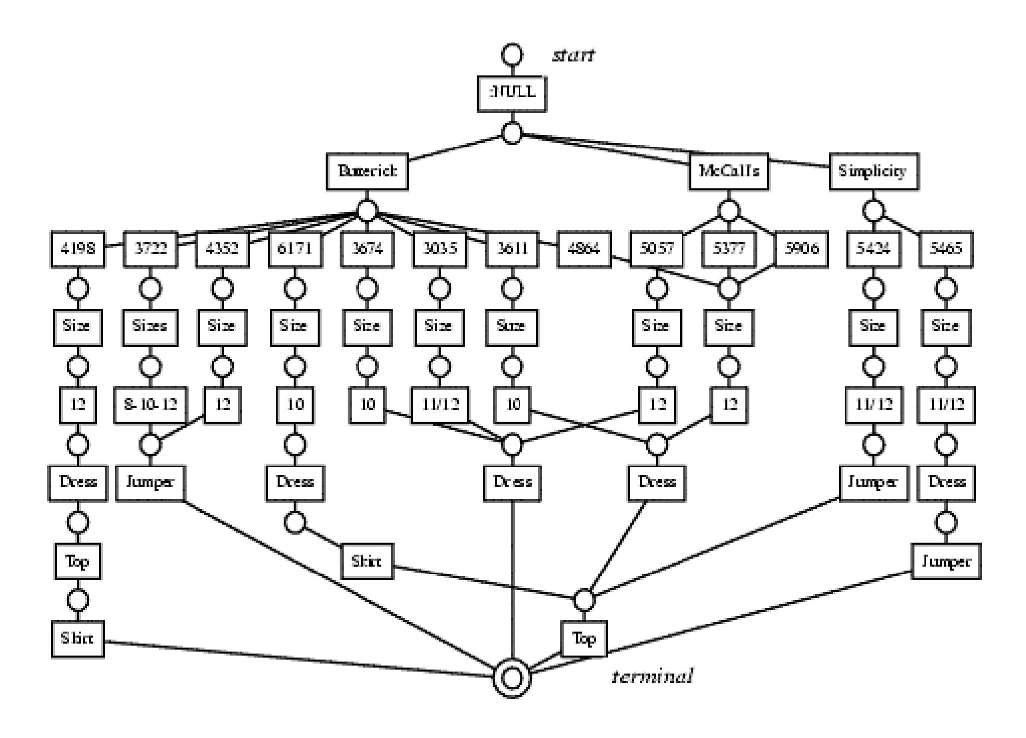


Abbildung 32 0-rekursiver Automat

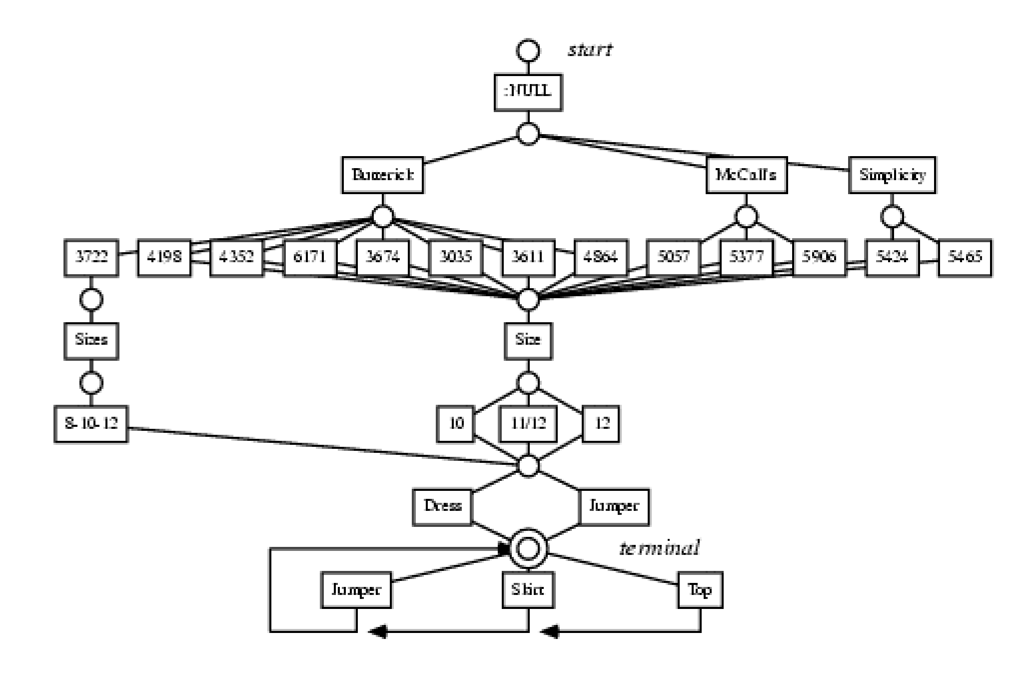


Abbildung 33 Automat nach Anwendung der Regeln von Schlimmer/Hermens

8.3.3.1.2.1 Regel 3a

Wie bereits angedeutet stellt die Regel 3a eine Modifikation der Angluin-Regel 2b dar. Schlimmer und Hermens verallgemeinerten das Kriterium der ursprünglichen Regel 2b von Angluin dahingehend, daß die betrachteten Zustände (S_x bzw. S_y) auch über ein *beliebiges* Token (A oder B) in den gemeinsamen Nachfolger-Zustand (S_1) übergehen können. Im konkreten Fall müssen außerdem die Label der Token T_1 und T_k übereinstimmen, die Zustände S_x und S_y also einen gemeinsamen 1-Vorgänger haben.

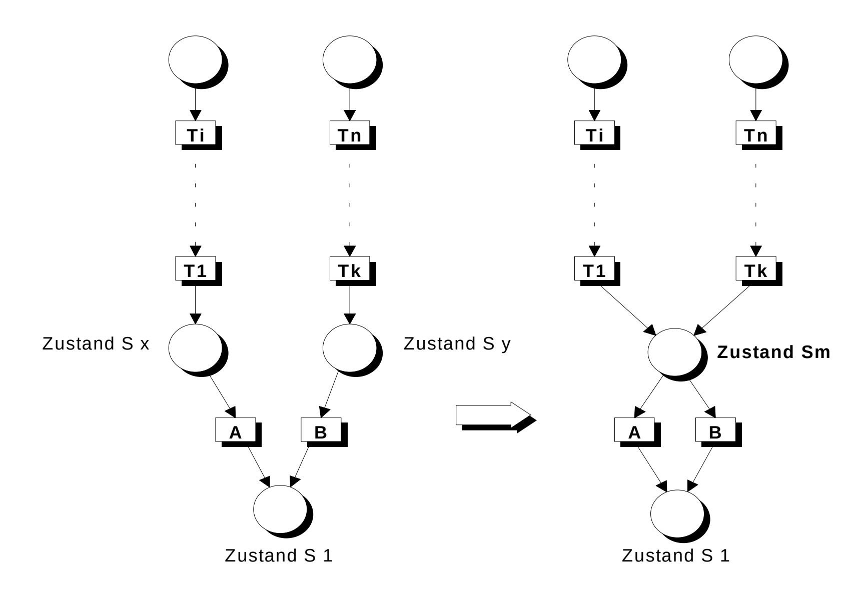


Abbildung 34 Mergingregel 3a

8.3.3.1.2.2 Regel 3b

Die Regel 3b (auch Kleene-Regel genannt) verallgemeinert den DFA durch Zyklenbildung. Die Tokensequenzen innerhalb der entstandenen Zyklen können dabei als optional betrachtet werden. Schlimmer und Hermens stellten diese in der graphischen Benutzeroberfläche (dem TCI) als Checkboxes dar, die maximal einmal ausgewählt werden konnten. Es kann jedoch auch sinnvoll sein einen Zyklus mehrfach zu durchlaufen. In diesem Fall ist im TCI ein Radiobutton als Darstellungselement vorgesehen. Der Zusammenhang zwischen dem DFA und der erzeugten Benutzeroberfläche wird in Kapitel 9 detailliert beschrieben.

Konkret erfüllen zwei Zustände S_x und S_y die Regel 3b falls sie identische k-Vorgänger besitzen und eine Transition beliebiger Länge zwischen diesen beiden Zuständen existie<rt. Abbildung 35 zeigt daß eine Transition, bestehend aus der Tokensequenz A,..,B, von Zustand S_x nach Zustand S_y besteht. Die Listen der Vorgänger- und Nachfolger-Token des verbleibenden Zustandes werden dabei wiederum um die entsprechenden Listen des zu löschenden Zustandes erweitert.

Als Ergebnis des Mergingvorganges kann die Tokensequenz A,..,B im TokenComponentInterface nun beliebig oft ausgewählt werden.

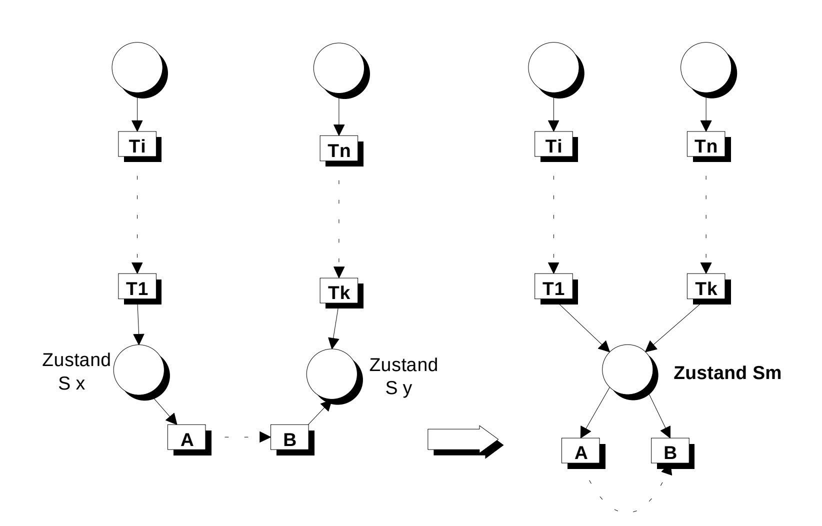


Abbildung 35 Mergingregel 3b

8.3.3.1.3 Eigene Erweiterung

8.3.3.1.3.1 Regel 3aE

Die Regel 3aE stellt im Vergleich zur ursprünglichen Regel 3a von Schlimmer und Hermens eine leichte Modifikation dar. Die Modifikation besteht v. a. in einer Abschwächung des Nachfolgerkriteriums der ursprünglichen Regel. Gemäß Regel 3aE werden nun zwei Zustände S_x und S_y zusammengefaßt, falls die jeweiligen k-Vorgänger übereinstimmen und die Zustände einen gemeinsamen Nachfolger-Zustand besitzen. Die Pfade vom jeweiligen Zustand bis zum gemeinsamen Nachfolger-Zustand müssen dabei disjunkt sein. Diese einschränkende Bedingung soll verhindern, daß Zustände deren Vorgänger-Zustände auf einem gemeinsamen Pfad liegen zusammengefaßt werden. Diese Regel kann im übrigen im Rahmen des Mergingvorganges alternativ zur Regel 3a ausgewählt werden.

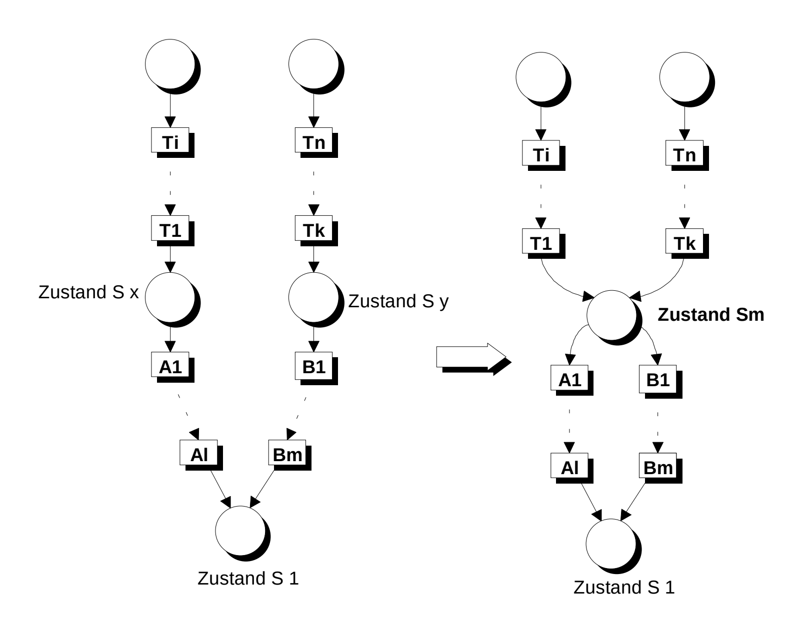
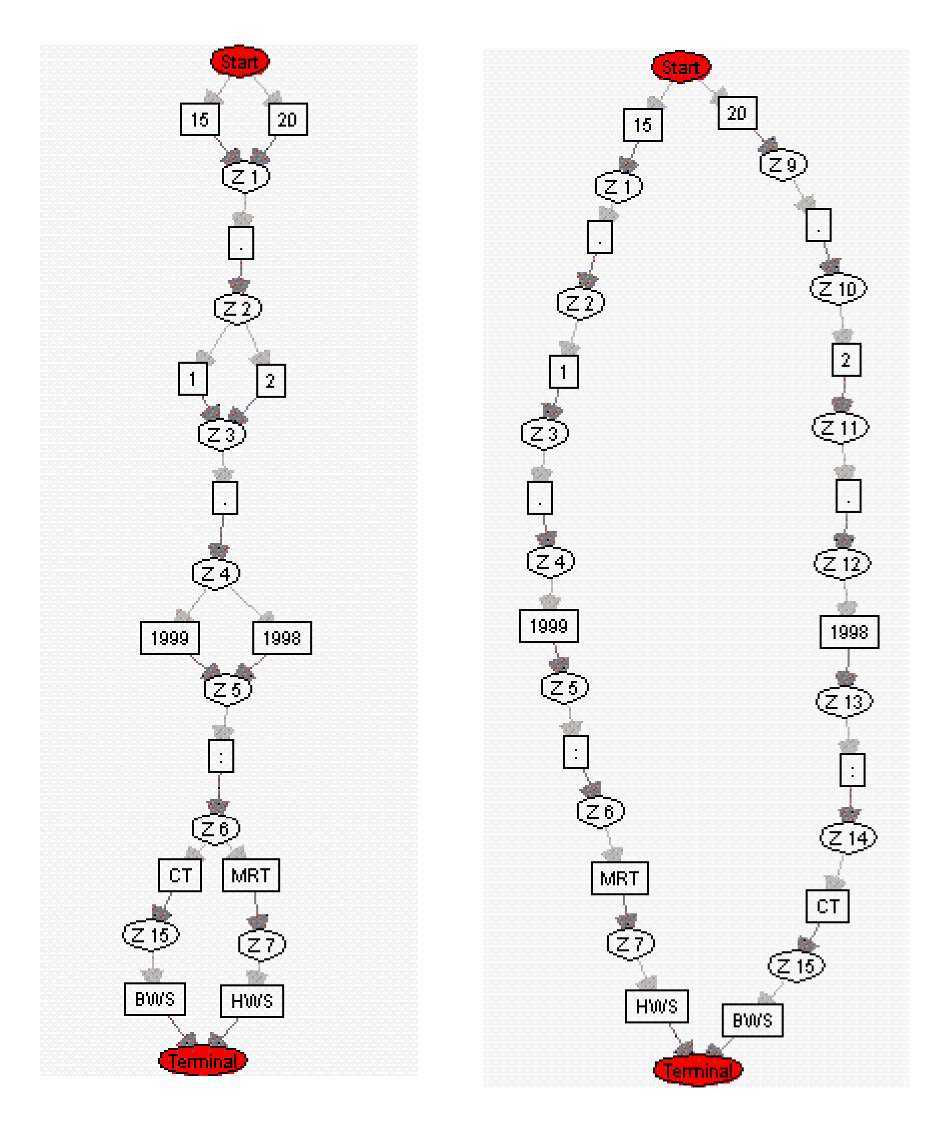


Abbildung 36 Mergingregel 3aE



Graph mit der erweiterten Version der Regel 3a

Graph mit der ursprünglichen Version der Regel 3a

Abbildung 37 Vergleich der ursprünglichen mit der erweiterten Regel 3a

Die Modifikation der Regel 3a von Schlimmer und Hermens soll im folgenden anhand zweier Befundfragmente erläutert werden:

1) 15.1.1999 : MRT HWS

2) 20.2.1998 : CT BWS

Die beiden Fragmente weisen folgende identische Struktur auf:

*⟨Tag⟩***.***⟨Monat⟩***.***⟨Jahr⟩* **:** *⟨Untersuchungsmethode⟩ ⟨untersuchter Wirbelsäulenabschnitt⟩*

Abbildung 37 zeigt die beiden Automaten, die vom System nach dem Einlesen der beiden Fragmente erzeugt wurden. In beiden Fällen wurden neben den Angluin-Regeln

1, 2a und 2b mit dem k-Vorgängerwert 0 die jeweilige Version der Regel 3a verwendet. Der von Schlimmer und Hermens vorgeschlagene Parameterwert für k wurde in beiden Fällen übernommen. Der Automat auf der linken Seite der Abbildung, der auf der Basis der erweiterten Regel 3aE erzeugt wurde gibt die Struktur der Fragmente deutlich besser wieder als der Automat auf der rechten Seite, bei dem die von Schlimmer und Hermens verwendete Regel 3a zugrunde gelegt wurde.

Entscheidend in diesem Fall sind die beiden Zustände Z6 und Z14 auf der rechten Seite der Abbildung. Zwar haben sie denselben 1-Vorgänger, nämlich „**:**“, jedoch kann die ursprüngliche Regel hier nicht angewendet werden, da sie keinen *direkten* gemeinsamen Nachfolger-Zustand besitzen. Neben der Regel 2a findet in diesem Fall keine weitere Regel Anwendung. Im Gegensatz dazu erfüllen die Zustände Z6 und Z14 die Bedingungen der erweiterten Regel 3aE und können folglich zusammengefaßt werden.

8.3.3.2 Übergeneralisierung

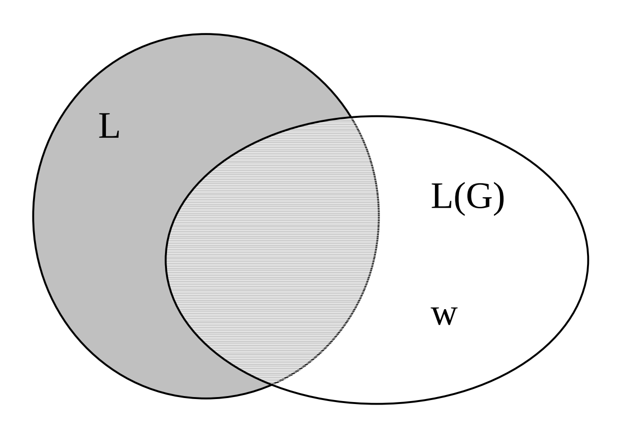


Abbildung 38 Übergeneralisierung

Die Abbildung 38 visualisiert das Prinzip der Übergeneralisierung. Formal liegt eine Übergeneralisierung der inferierten Grammatik G bzw. des entsprechenden endlichen Automaten vor, falls mindestens ein $w \in \Sigma^*$ existiert, so daß $w \in L(G)$ und $w \notin L$. Konkret bedeutet dies, daß die Sprache der inferierten Grammatik L(G) Strings enthält, die nicht Teil der gesuchten Sprache L sind. Da der String w ein negatives Beispiel in Bezug auf die gesuchte Sprache L ist und L(G) diesen String enthält ist L(G) somit inkonsistent mit der gegebenen Beispielmenge.

Im folgenden sollen an einem konkreten Beispiel die Auswirkungen von Übergeneralisierungen verdeutlicht werden. Die Komplexität des Beispieles ist bewußt gering gehalten um die erzeugten Automaten in Abbildung 39 übersichtlich zu gestalten.

Gegeben sei die folgende Menge positiver Beispiele:

1. *Chronisches Zervikalsyndrom*

2. Zervikalsyndrom

3. Status vor Microtherapie

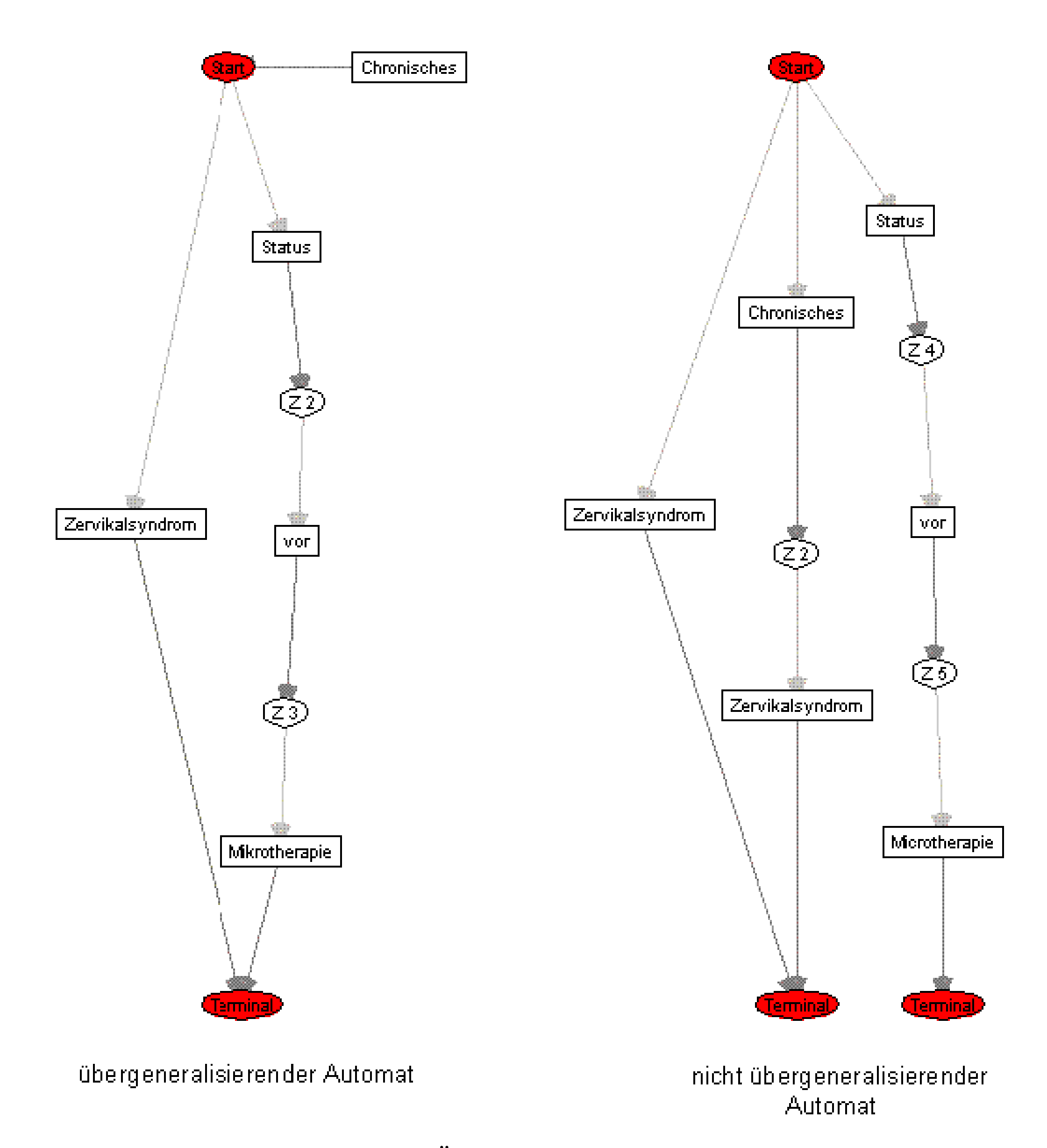


Abbildung 39 Übergeneralisierender Automat

Die in Abbildung 39 dargestellten Automaten wurden auf der Grundlage der Angluin-Regeln 1, 2a und 2b mit unterschiedlichen Parameterwerten k erzeugt. Auf der linken Seite der Abbildung wurde k auf 0 gesetzt, d. h. es wird vermutet, daß die gesuchte Sprache 0-reversibel ist. Das Ergebnis zeigt, daß diese Hypothese inkorrekt war, da der Automat als Folge einer Übergeneralisierung den ungrammatikalischen Satz

„*Chronisches Status vor Microtherapie*“ erzeugt. Auf der rechten Seite sieht man einen Automaten bei dem der Parameter für k um eins erhöht wurde. Dieser inferierte Automat erzeugt ausschließlich grammatikalisch sinnvolle Sätze.

Um entscheiden zu können ob der inferierte Automat übergeneralisiert, ist die Ermittlung der Sprachklasse der Befunddaten nötig, da im Vorfeld des Lernverfahrens keine Angaben über die Sprachklasse der Befunddaten bestehen. Zur Ermittlung des Parameters k wurde folgendes Verfahren von [Berwick et al. 1987] verwendet:

```
Annahme : alle Strings außerhalb eines bestimmten endlichen Korpus
          sind negative Beispiele
Setze k=0
Wiederhole
   Falls
     der DFA alle positiven Beispiele des Korpus
     und keine negativen abdeckt
   ende
   Sonst
     Erhöhe k um 1
```

Tabelle 7 Verfahren zur Ermittlung des Parameters k

Sowohl die Wahl des k-Vorgängerwertes der jeweiligen Regel als auch die Zusammensetzung der Regelmenge beeinflußt entscheidend die Qualität des erzeugten DFA im Hinblick auf die Benutzerfreundlichkeit des entsprechenden Befundungssystems. Tendenziell kann festgestellt werden, daß mit steigenden Parameterwerten (größere Anzahl der betrachteten Vorgänger eines Zustandes) die Zahl der Übergeneralisierungen abnimmt und somit die Möglichkeit ungrammatikalische Sätze zu erzeugen. Eine ausführlichere Diskussion der Auswirkungen unterschiedlicher Parameter- bzw. Regelkombinationen findet sich in Kapitel Testergebnisse.

9 Das TokenComponent-Interface

Das TokenComponent-Interface (abgekürzt TCI) ist die graphische Benutzeroberfläche, die es einem Nutzer erlaubt, aus einmal gelernten Texten weitere Texte jeweils auf eine effiziente Art zu erstellen. Insbesondere wird der Benutzer strukturiert durch den Erstellungsvorgang geleitet. Ärzte, vor allem Radiologen, unterstützt *mAGENTa* so aktiv bei der Befunderfassung, indem sie schrittweise durch den Befundungsvorgang geleitet werden.

Da das TokenComponent-Interface sowohl theoretische, als auch praktische Bewandtnis hat, wird in diesem Kapitel auf die theoretischen Hintergründe eingegangen, während im Benutzerhandbuch in dem Kapitel über das TCI die praktische Anwendung durch den Benutzer im Vordergrund steht.

9.1 Zusammenhang zwischen DFA und TokenComponent-Interface

Das TCI ist eine direkte Abbildung des gelernten DFA in eine benutzerfreundliche Darstellung.

Ein einzelner Text kann als Pfad betrachtet werden, der durch den DFA beschritten wurde. Von einem Zustand aus kann jeweils ein Wort (Token) gemäß der Zustandsüberführungsfunktion gewählt werden, um in einen nächsten Zustand zu gelangen. Beginnend bei dem StartZustand, manövriert der Benutzer so von Zustand, zu Zustand bis er in einem TerminalZustand endet. Die Konkatenation (Verkettung) der einzelnen Wörter, die jeweils einen Zustand in den nächsten überführt haben, ergibt nun einen zusammenhängenden Text.

Für jeden Zustand im DFA, der mehr als einen Zustandsübergang hat (also mehr als zwei ausgehende Kanten) kann der Benutzer sich entscheiden, welchen Weg er gehen möchte. Er hat also die Wahl, welches Wort er eingeben möchte, um in den nächsten Zustand zu gelangen. Diese Wahlmöglichkeit an einem Zustand wird im TCI durch TokenButtons dargestellt. Jeder TokenButton steht für eine ausgehende Kante des Zustandes. Der TokenButton wird demnach mit dem Wort beschriftet, das gemäß der Zustandsübergangsfunktion den Zustand in einen Folgezustand überführt.

Die folgende Abbildung zeigt exemplarisch die graphische Ansicht des gelernten DFA für den MetaAbschnitt *Koronare Risikofaktoren*, nachdem 13 typische Beispiele aus der Domäne der EBT-Befunde gelernt worden sind.

Eine genaue Beschreibung der einzelnen Elemente des Fenster sind in Kapitel 11.5.3 "Der Graph" zu finden. Hier genügt es zu wissen, daß Kreise Zustände repräsentieren und Rechtecke Tokens. Der rote Startzustand am oberen Rand des Fensters repräsentiert

den MetaZustand für den Abschnitt *Koronare Risikofaktoren*. Der rote TerminalZustand am unteren Rand den folgenden MetaZustand.

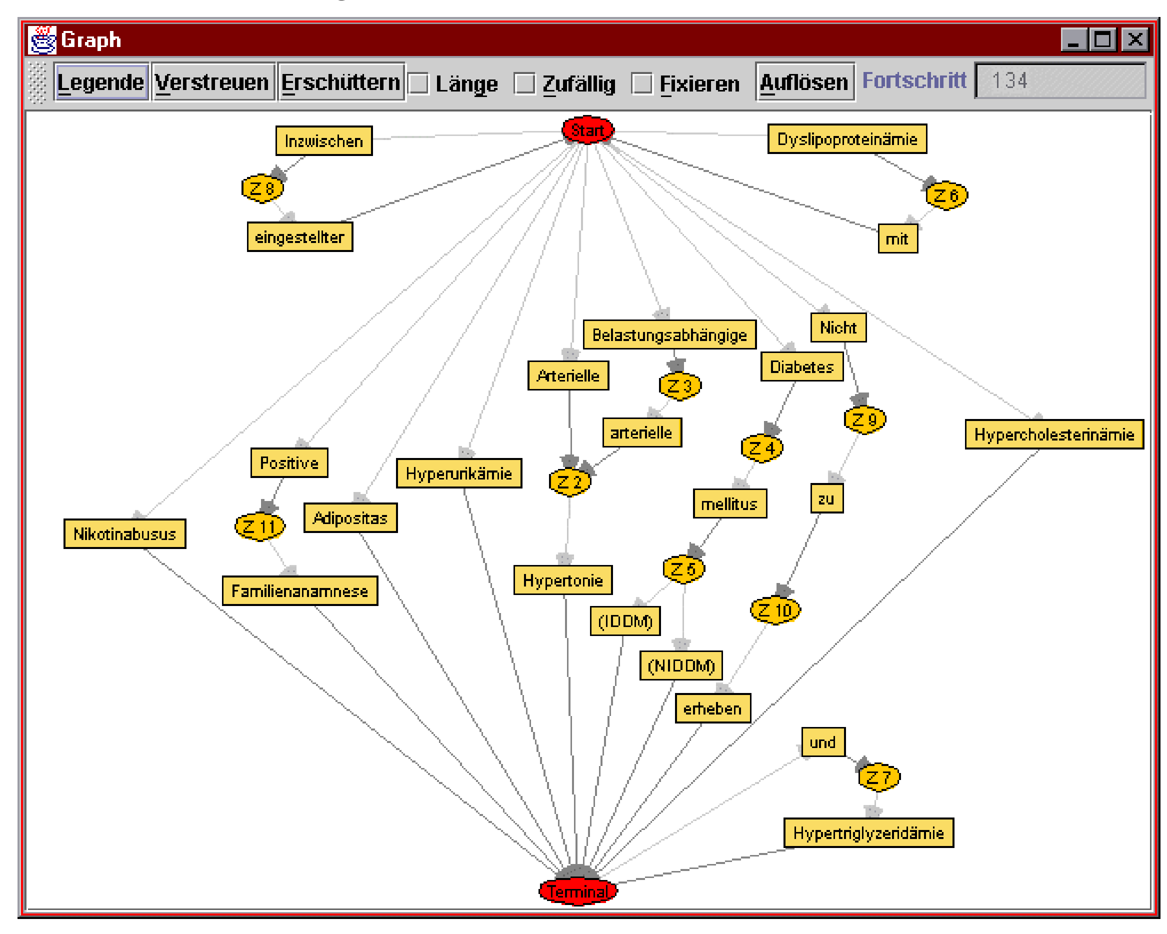


Abbildung 40 Graph für Koronare Risikofaktoren

Als Beispiel dient der in seiner Graph-Darstellung abgebildete DFA.

9.2 ZustandsRahmen

Ein ZustandsRahmen repräsentiert einen Zustand, von dem mehrere Kanten ausgehen. Es enthält alle TokenComponents, die diesen Zustand repräsentieren. Zuerst die TokenButtons und evtl. TokenChoices und als letzte dargestellte Komponente eine EmptyComponent.

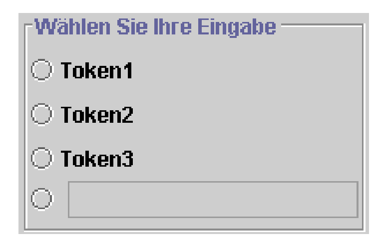


Abbildung 41 ZustandsRahmen

Die ausgehenden Kanten eines Zustandes sind jeweils mit Tokens beschriftet. Jedes dieser Tokens (hier die Tokens *Token1*, *Token2* und *Token3*) wird durch eine eigene Komponente dargestellt. Diese Tokens überführen den Zustand jeweils in einen Nachfolgezustand. Oftmals ist es in den gelernten Texten so, daß lineare Pfade dem Zustand folgen, für den der Rahmen generiert wird. Die Worte auf diesen Pfaden werden konkateniert und an das erste Token gehangen.

Als letzte Komponente in einem ZustandsRahmen befindet sich immer eine EmptyComponent, um neue Eingaben des Benutzers aufnehmen zu können, die durch die bisherigen Komponenten nicht abgedeckt worden sind.

Die einzelnen TokenButtons können optional der Wahrscheinlichkeit nach geordnet werden, mit der sie bei künftigen Befundungen angeklickt werden. Dies wird aus den Frequenzen der Tokens, die durch die einzelne TokenComponent repräsentiert werden, ermittelt. Die Frequenz eines Tokens gibt an, wie oft bisher dieses Token in Befunden vorkam bzw. wie oft es schon vom Benutzer angeklickt worden ist, um in einen neuen Befund aufgenommen zu werden.

Für die wahrscheinlichste TokenComponent wird jeweils als Vorschau ein weiterer Rahmen für den Zustand angezeigt, der erreicht wird, wenn diese TokenComponent betätigt wird.

Die einzelnen TokenComponents im ZustandsRahmen bilden eine TokenComponent-Auswahl. Dies bedeutet, daß zu einem Zeitpunkt nur eine Komponente angewählt werden kann. Wenn eine angewählt worden ist, so wird die Beschriftung der Komponente in den Befundungstext übernommen, und für den nachfolgenden Zustand der Komponente der zugehörige ZustandsRahmen angezeigt, um hier mit der Eingabe fortzufahren.

9.3 Die TokenComponents

Zur graphischen Darstellung von Tokens gibt es drei verschiedene TokenComponents: die TokenButtons, die TokenChoices und die EmptyComponents. Wird eine von ihnen betätigt, so wird für den Zustand der über das letzte dargestellte Token erreicht wird ein neuer ZustandsRahmen angezeigt, der anstelle des alten angezeigt wird.

9.3.1 TokenButton

Ein TokenButton repräsentiert ein Wort bzw. Token (hier *Token1*) und eventuelle Listen von vorhergehenden und nachfolgenden Tokens, die zusammengefaßt worden sind.



Abbildung 42 TokenButton

Da in den gelernten DFA häufig Pfade vorkommen auf denen die Zustände nur durch ein einziges Wort in einen Nachfolgezustand überführt werden können, ist es sinnvoll, diese auch nur durch einen einzigen TokenButton darzustellen. Würde für jeden dieser Worte ein eigener TokenButton generiert werden, müßte der Benutzer jedes Wort seines Textes einzeln anklicken, was einer benutzerfreundlichen Darstellung widerspräche.

So werden die Worte auf linearen Pfaden, die vor bzw. hinter einem Zustand mit einem Verzweigungsgrad größer als eins liegen „aufgesammelt" und vor bzw. hinter die Beschriftung des TokenButtons geschrieben.

9.3.2 TokenChoice

Für eine bessere Übersichtlichkeit werden TokenButtons mit gleichem Start- und Endzustand ab einem einstellbaren Schwellenwert in einer Aufklappliste, der sogenannten TokenChoice, zusammengefaßt.

Die Anzahl, ab wann TokenButtons zusammengefaßt werden sollen, ist voreingestellt auf 5 (hier 3), aber beliebig im ganzzahligen Bereich änderbar. Die TokenChoice dient der Übersichtlichkeit, damit nicht zu viele Komponenten gleichzeitig dargestellt werden.

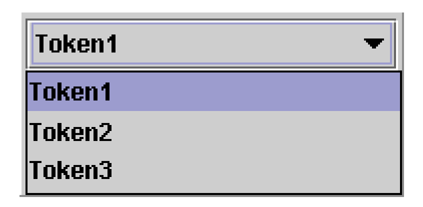


Abbildung 43 TokenChoice

9.3.3 EmptyComponent

Da die gelernten Befunde evtl. nicht alle Fälle abdecken, ist es wichtig, daß der Benutzer einfach Ergänzungen am DFA vornehmen kann, indem er einen weiteren Text eingeben kann, ohne einen kompletten Befund eingeben zu müssen.



Abbildung 44 EmptyComponent

Für diese gezielte Modifikation wird dem Benutzer im TCI eine graphische Komponente geboten, die als EmptyComponent bezeichnet wird. Eine EmptyComponent repräsentiert im eigentlichen Sinne keine Tokens des DFA, sondern dient dazu, neue Tokens aufzunehmen, die vom Benutzer gezielt eingegeben werden können. Dafür enthält sie ein Textfeld für Ergänzungen.

Diese werden direkt in den DFA integriert, und zwar hinter den Zustand, den der ZustandsRahmen repräsentiert, in dem sich die EmptyComponent befindet. Hierdurch wird dem Benutzer eine flexible Möglichkeit gegeben, das Lernergebnis zu erweitern, wenn z.B. ein Krankheitsbild über zusätzliche Symptome verfügt, die bisher noch nicht vorgekommen sind.

9.4 Algorithmus zur Generierung des TokenComponent-Interfaces

```
Beginne beim StartZustand

Solange der betrachtete Zustand genau ein NachfolgeToken hat

      merke das Token in einer Liste

      wenn der NachfolgeZustand des Tokens ein MetaZustand ist: beende
      Schleife

      der NachfolgeZustand wird zum betrachteten Zustand

=> Eine lineare Liste von Token (Vorgängerliste)

erstelle für den zuletzt besuchten Zustand einen leeren ZustandsRahmen

ist der letzte Zustand Terminal- oder Meta-Zustand

      dann erstelle TokenButton für VorgängerListe

      sonst sammle NachfolgerListen für jeden Nachfolger des letzten
Zustandes

gibt es Nachfolge-Token, die den selben Folge-Zustand haben, und ist
deren Anzahl größer als der Parameter für die TokenChoice,

      dann trage die VorgängerToken-Liste, die Tokens, und die
      aufgesammelte NachfolgerToken-Liste in eine TokenChoice ein

trage die übrigen Token

      jeweils mit der VorgängerToken-Liste und deren gesammelter
      NachfolgerToken-Liste in einen TokenButton ein

erstelle eine EmptyComponent

füge alle TokenComponents geordnet nach der Frequenz ihrer Token in
das ZustandsRahmen

beginne mit dem NachfolgeZustand der ersten TokenComponent wieder von
vorne
```

Tabelle 8 Algorithmus zur Generierung des TokenComponent-Interfaces

9.5 Die Generierung des TCI an einem Beispiel

Die folgende Abbildung zeigt das TCI des in Abbildung 40 "Graph für koronare Risikofaktoren" dargestellten MetaAbschnittes.

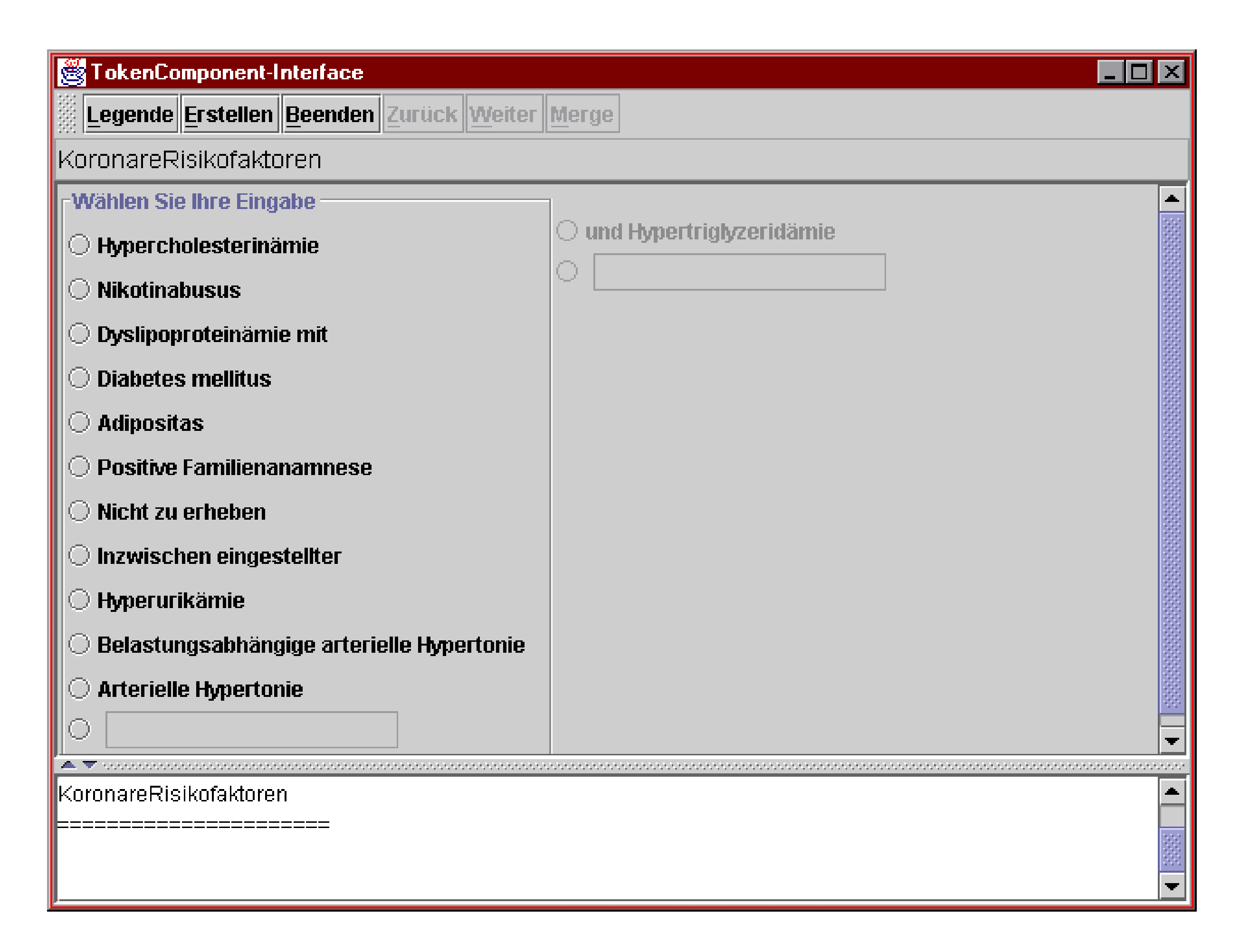


Abbildung 45 TokenComponent-Interface für Koronare Risikofaktoren

Beginnend bei dem MetaZustand mit der Aufschrift *Start* werden dessen ausgehende Kanten betrachtet. Jede dieser 11 Kanten wird durch einen TokenButton repräsentiert.

Einige der Kanten bilden lineare Pfade, deren Worte zusammengefaßt wurden. Beispielsweise werden die Worte auf dem Pfad *Start* über *Z 4* zu *Z 5* aufgesammelt und zu der Beschriftung *Diabetes mellitus* zusammengefaßt. Erst nach dem Zustand mit der Aufschrift *Z 5* findet eine Verzweigung statt, die, falls der Benutzer diesen TokenButton wählen sollte, zur Generierung von neuen TokenButtons führt.

Vier der TokenButtons, die den MetaZustand repräsentieren, stehen also für ein einzelnes Token (*Hypercholesterinämie*, *Nikotinabusus*, *Adipositas und Hyperurikämie*). Die übrigen sieben stehen für zwei oder mehr linear aufeinander folgende Tokens.

In der Abbildung des TCI ist zu sehen, daß auf den TokenButton mit der Beschriftung *Hypercholesterinämie* ein neuer Rahmen mit dem TokenButton *und Hypertriglyceridämie* folgt.

Der Titel des Abschnittes findet sich als Überschrift oberhalb der TokenButtons. Im unteren Bereich des Fensters werden die Beschriftungen der nacheinander angeklickten TokenButtons sowie die Abschnittstitel in einem Textfeld dargestellt. Solange der Folgezustand eines TokenButtons kein MetaZustand oder Terminalzustand ist, werden für diesen die weiteren TokenButtons dargestellt. Wird schließlich ein Terminalzustand erreicht, so ist der Befundungsvorgang beendet und der Benutzer kann den erstellten Befund direkt abspeichern oder ausdrucken.

In der Arbeit von Schlimmer & Hermens [Schlimmer et al. 1993] wurde ein Auswahlfeld neben den TokenButtons als Eingabekomponente vorgestellt. Diese würde angeben, daß sich der Zustand in einem Zyklus befindet. Mittels des Auswahlfeldes kann der Zyklus genau einmal durchlaufen werden oder eben nicht. Oftmals ist es aber sinnvoll, einen Zyklus mehrfach zu durchlaufen, dies kann durch mehrfaches Betätigen eines TokenButton dargestellt werden.

9.6 Die Parametrisierungen

Mittels der Parametrisierungen des TokenComponent-Interface lassen sich Einstellungen vornehmen, die bei der Benutzung des TCI in der Oberfläche Informationen über den DFA anzeigen, um den Benutzter bei der Erstellung von Texten behilflich zu sein. Durch weitere Einstellungen läßt sich Einfluß auf den Weg des Benutzers durch den DFA nehmen, und somit auch auf den erstellten Text, der durch die Tokens auf dem Pfad durch den DFA gebildet wird.

9.6.1 Anzeige von Nachfolgersymbolen

Ist dieses Auswahlfeld selektiert, so werden im TCI Informationen über denjenigen Zustand angezeigt, der erreicht wird, wenn eine TokenComponent betätigt wird.

Die Eigenschaften dieses Zustandes werden jeweils durch ein bestimmtes Symbol dargestellt.

Eigenschaft des nachfolgen Zustandes	Symbol
TerminalZustand	-■
StartZustand	-S
MetaZustand	-M
Zustand befindet sich in einem Zyklus	-o

Tabelle 9 Symbole für die Eigenschaften des nachfolgenden Zustandes

Diese Symbole dienen zur Information des Benutzers und ermöglichen ihm, die nächste Auswahl der TokenComponents differenzierter zu treffen.

Da ein Zustand auch über mehrere der folgenden Eigenschaften verfügen kann, werden die entsprechenden Symbole nebeneinander von rechts nach links angezeigt. Befindet sich z.B. ein MetaZustand in einem Zyklus, so wird das MetaZustand-Symbol am rechten Rand der TokenComponent gezeichnet, das Symbol für den Zyklus links daneben.

9.6.2 Anzeige von Steuerzeichen

Texte sind im Allgemeinen durch Steuerzeichen formatiert. Am Ende eines Satzes oder Abschnittes finden sich diese speziellen Zeichen, die dafür sorgen, daß z.B. die Schreibmarke in die nächste Zeile gesetzt wird. Da diese Steuerzeichen in Texten sehr häufig vorkommen, aber keinen semantischen Vorteil zeigen, sind sie in der Ansicht des TCI eher hinderlich, da sie den Blick auf den eigentlichen Text der TokenComponents ablenken.

Deshalb sind diese Steuerzeichen gemäß den Voreinstellungen nicht sichtbar. Sollte dieses Auswahlfeld selektiert sein, werden im TCI die Steuerzeichen (auch Escape-Sequenzen genannt) in der Standardnotation der C-Escape-Sequenzen dargestellt.

Steuerzeichen	Escape-Sequenz	Unicode-Wert
Tabulator (horizontal)	\t	/u0009
Zeilenvorschub	\n	/u000a
Wagenrücklauf	\r	/u000d

Tabelle 10 Sonderzeichen / Escape-Sequenzen

9.6.3 Mehrfaches Besuchen von MetaZuständen

Folgt der Lernprozeß dem Konzept des deskriptiven Mergings der MetaZustände (siehe Kapitel 8 "DFA-Generierung"), so ist es möglich, daß die Struktur der gelernten Texte einen Zyklus aufweist. Dies hat zur Folge, daß ein MetaZustand mehrfach besucht werden kann. Ist dies nicht gewünscht, d.h. die MetaZustände sollen nur genau einmal besucht werden können, so läßt sich auch dies einstellen.

Wird unter dieser Konfiguration bei Eingabe in das TCI ein MetaZustand ein zweites Mal besucht, so wird automatisch der Rahmen für den nächsten noch nicht besuchten MetaZustand generiert.

Der nächste MetaZustand wird durch einen DFS-Durchlauf ermittelt, der bei dem erneut besuchten MetaZustand beginnt. Sollte sich in dem durch den DFS ergebenden Spannbaum kein noch nicht besuchter MetaZustand befinden, so erfolgt die weitere Suche ab dem StartZustand des gesamten DFA. Sollte auch hier kein unbesuchter

MetaZustand gefunden werden, so ist der Befundungsvorgang hiermit abgeschlossen, da alle MetaZustände und folglich auch alle Abschnitte einmal besucht worden sind.

Ist diese Option selektiert, so kann ein MetaAbschnitt, dessen MetaZustand sich in einem Zyklus befindet, beliebig oft durchlaufen werden.

9.6.4 Mehrfache Anzeige von MetaTiteln

Ist die vorige Option selektiert, d.h. ein MetaZustand in einem Zyklus kann mehrfach besucht werden, so wird bei jedem Besuch des MetaZustandes im TCI der Titel des MetaZustandes in den Befund übernommen, um den Beginn eines neuen MetaAbschnittes zu signalisieren.

Dies kann von dem Benutzer nicht erwünscht sein und mittels dieser Option verhindert werden. Ist sie selektiert, so wird der Titel des MetaZustandes nur einmal in den Befund aufgenommen und auch bei erneutem Besuch nicht mehr hinzugefügt.

9.6.5 Sortierung der Tokens nach Frequenz

Das TCI präsentiert dem Benutzer die einzelnen TokenComponents in der Reihenfolge, die für den weiteren Befundungsvorgang am wahrscheinlichsten ist. Diese Reihenfolge ergibt sich gemäß der Annahme, daß Textesequenzen, die in der Vergangenheit häufig in gelernten Texten vorkamen, auch in der Zukunft häufiger erfaßt werden.

Jedes Token verfügt über eine Frequenz, die angibt, wie häufig dieses Token in den bisher gelernten Texten auftrat. Diese Frequenz wird neben dem Lernprozeß auch dann erhöht, wenn die TokenComponent, die dieses Token repräsentiert, während der Benutzung des TCI betätigt wird. Durch häufiges Betätigen einer TokenComponent wird diese in der Anzeige des TCI nach oben wandern.

Je höher die Frequenz ist, desto höher ist die Wahrscheinlichkeit für ein Auftreten dieses Tokens nach seinem Vorgängerzustand. Demzufolge ergibt sich die Sortierung der TokenComponents als Sortierung der Frequenzen des jeweils ersten Tokens in den TokenComponents.

Sollte eine Sortierung nach der Frequenz der Tokens nicht gewünscht sein, so ist dieses Auswahlfeld zu deselektieren. Die Frequenzen werden somit nicht mehr bei der Reihenfolge der TokenComponents berücksichtigt. Sie werden in der Reihenfolge dargestellt, die der Lernalgorithmus für die Zustände und Token vorgibt. Werden keine weiteren Text gelernt, so ändert sich auch nicht die Reihenfolge der TokenComponts. Selbst die Erstellung eines Textes mit Hilfe des TCI verändert die Reihenfolge der TokenComponents nicht, obwohl die Frequenz der Token jeweils erhöht wird. Erst wenn dieses Auswahlfeld wieder selektiert ist, werden die TokenComponents automatisch in der sortierten Reihenfolge dargestellt.

Die Option, die Sortierung zu deaktivieren, ist insbesondere für diejenigen Benutzer gedacht, die über Erfahrung im Umgang mit *mAGENTa* verfügen. Durch häufige

Erstellung von Texten mit dem TCI hat der Benutzer sich eingeprägt, an welcher Position sich eine bestimmte TokenComponent befindet. Erfahrenen Benutzern genügt es, nur die ersten Tokens oder auch nur die ersten Buchstaben einer TokenComponent zu lesen, um zu wissen, wie die komplette Beschriftung heißt. Dies führt zu einer Erhöhung des Geschwindigkeit, mit der ein Text erfaßt werden kann. Es kann dann unerwünscht sein, durch Erfassung weiterer Texte die Position der TokenComponents zu verändern.

9.6.6 Protokollder Positionen der TokenComponents

Mit dieser Einstellung wird festgelegt, daß die Position einer angeklickten TokenComponent innerhalb des TCI während der Erfassung eines Textes protokolliert werden soll. Am Ende eines Erfassungsvorganges wird eine Auswertung darüber angezeigt, an wievielter Position sich eine TokenComponent innerhalb der Auswahl befand. Dies ist für die statistische Auswertung interessant, um zu überprüfen, wie gut die Sortierung der TokenComponents den Erfassungsvorgang unterstützt.

9.6.7 Anzahl TokenChoice

Es kann festgelegt werden, wie viele TokenButtons mit gleichem Start- und Endzustand zu einer TokenChoice zusammengefaßt werden sollen.

Gehen von einem Zustand mindestens so viele Kanten aus, wie festgelegt worden ist, und enden alle von diesem Zustand ausgehenden lineare Pfade in dem selben Zustand, so wird eine TokenChoice erzeugt. Die konkatenierten Tokens, die jeweils auf einem ausgehenden Pfad liegen, bilden einen Eintrag der TokenChoice.

Wird ein Wert angegeben, der größer ist als der maximale Verzweigungspfad des DFA, so werden nur TokenButtons generiert.

10 Erzeugen der Befunddokumente

10.1 Aufbau des Befunddokumentes

Bei der Texterfassung mittels des TokenComponent-Interfaces betätigt der Benutzer eine Folge von TokenComponents. Beginnend mit dem StartZustand werden die einzelnen Zustände, die durch die ZustandsRahmen repräsentiert werden, gemäß der Zustandsübergangsfunktion in andere Zustände überführt. Dieser Vorgang führt schließlich zu einem TerminalZustand. Ist dieser erreicht, so ist der Texterfassungsvorgang beendet, und der endgültige Text kann erstellt werden.

Der erfaßte Text ergibt sich jetzt aus den Tokens, die auf dem Pfad liegen, den der Benutzer durch die TokenComponents ausgewählt hat.

Zusätzlich zu diesem erfaßten Text gibt es die Möglichkeit, einen Briefkopf und einen Brieffuß vor bzw. hinter dem Text einzufügen. In diesen Texten können sich beispielsweise ein Standardanschreiben und eine Verabschiedungsformel befinden.

Ein typischer radiologischer Befundtext, so wie er in der täglichen Praxis auftritt, ist in Kapitel 3.2.2 "Der radiologische Befund" angegeben. Hier ist deutlich die oben dargelegte Dreiteilung des Textes zu erkennen.

Briefkopf und Brieffuß sollten einfache Textdateien sein, die als Epilog bzw. Prolog zu dem erfaßten Text dienen. In ihnen können Schlüsselworte eingefügt sein, die für bestimmte Angaben, die in allen Texten immer wieder vorkommen, Platzhalter sind. Beispielsweise ist hier an das aktuelle Datum, die aktuelle Uhrzeit, das Kürzel des befundenden Arztes, usw. zu denken. Einige können automatisch von *mAGENTa* ersetzt werden, andere müssen für jeden Text individuell eingetragen werden.

Nach der Fertigstellung des gesamten Textes durch Hinzufügen von Biefkopf und Brieffuß kann dieser gespeichert oder direkt ausgedruckt werden.

Die folgenden Abschnitte zeigen exemplarisch einen typischen Briefkopf und Brieffuß, sowie eine Aufstellung von Platzhaltern und Schlüsselworten, die bei der Verwendung von *mAGENTa* eingesetzt werden können.

10.2 Briefkopf

Ein einfacher Briefkopf könnte z.B. in folgender Form gestaltet sein:

```
                        Lehrstuhl für Radiologie und MikroTherapie
                        Prof. Dr. med. Dietrich H.W. Grönemeyer
                        Institut für MikroTherapie
                        Universitätsstraße 142
                        D - 44799 Bochum
                        Telefon 0234 / 9 71 31 - 0
                        Telefax 0234 / 9 71 31 - 99
                        e-mail info@microtherapy.de
                        Internet www.microtherapy.de

ANSCHRIFT
ANSCHRIFT
ANSCHRIFT
ANSCHRIFT

                        Bochum, # DATE ## , # TIME ##


Sehr geehrte ANREDE,
```

Tabelle 11 Briefkopf

10.3 Brieffuß

Ein einfacher Brieffuß, der hinter dem durch das TCI erfaßten Text steht, könnte beispielsweise folgende Form haben:

```

Mit freundlichen Grüßen


NAME
```

Tabelle 12 Brieffuß

10.4 Schlüsselworte

`ANSCHRIFT` und `ANREDE` sind Platzhalter, die von dem Arzt entsprechend der Patientendaten gefüllt werden müssen.

`# DATE ##` und `# TIME ##` sind besondere Schlüsselworte, die von *mAGENTa* automatisch durch das aktuelle Datum und die aktuelle Zeit ersetzt werden.

Der Platzhalter `NAME` kann jeweils durch den Namen des behandelnden Arztes ersetzt werden. Arbeitet ein Arzt immer mit demselben Brieffuß, so kann der Name natürlich auch direkt in den Brieffuß eingetragen werden.

11 Benutzerhandbuch

In diesem Kapitel wird die Benutzung des *mAGENTa*-Systems, so wie es schließlich realisiert worden ist, anschaulich dargestellt. Auch ohne die theoretischen Vorkenntnisse der Kapitel aus dem ersten Teil der Diplomarbeit sollte es dem ungeübten Nutzer möglich sein, *mAGENTa* mittels der intuitiv zu bedienenden Oberfläche zu benutzen.

Leichte Bedienbarkeit war bei dieser interdisziplinären Arbeit eine Hauptaufgabe. Eine effiziente Befundeingabe für den Arzt sollte durch das System ermöglicht werden. Die umfangreiche Parametrisierung dient zusätzlich dazu, das System den Wünschen des Benutzers entsprechend anzupassen.

Bei der Beschreibung der einzelnen Komponenten des Systems wurde darauf geachtet, ein durchgängiges Beispiel aus dem realen Alltag eines Radiologen zu verwenden. Es wurden reale Befundtexte zu EBT-Aufnahmen des Herzens verwendet.

11.1 Systemvoraussetzungen

mAGENTa ist gemäß des Client/Server Paradigmas aufgebaut [Comer 1999]. Der Server stellt das System als Dienst zur Verfügung; der Client greift zur Benutzung dieses Dienstes auf den Server zu. Client und Server nutzen zur Kommunikation den TCP/IP Protokollstapel.

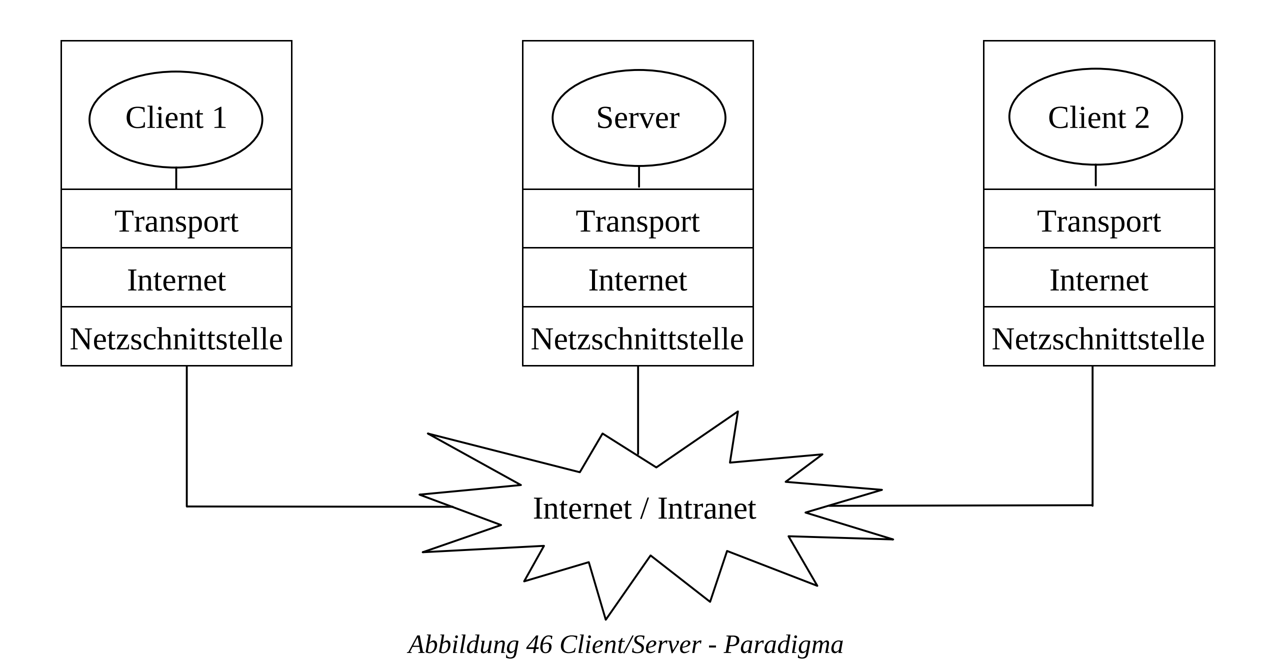


Abbildung 46 Client/Server - Paradigma

Sowohl der Client als auch der Server müssen über einen Zugang zum selben Netzwerk verfügen (Internet oder Intranet), in dem mit *mAGENTa* gearbeitet werden soll. Dieser Zugang läuft wie aus der obigen Abbildung ersichtlich über eine Netzschnittstelle (Netzwerkkarte, Modem oder dergleichen), die Internet-Protokolle (IP) und die Transportprotokolle (TCP). Daten werden über diese einzelnen Schichten zwischen der Anwendung auf dem Client und der Anwendung auf dem Server übermittelt.

11.1.1 Der Client

Um auf einzelne Seiten, welche auf entfernten Servern in einem Netzwerk stehen, zugreifen zu können, wird ein Browser benötigt, der diese Seiten lesen und interpretieren kann. Die Seiten im Internet sind mittels der Hypertext Markup Language (HTML) beschrieben. HTML ist eine Art Seitenbeschreibungssprache, die Text, Formatierungen und Verweise auf weitere Seiten, bzw. andere Objekte (z. B. Bilder, Töne, Videos, Java-Applets) bietet. Der Browser liest über das Netzwerkprotokoll eine Seite, die der Server anbietet, ein und verarbeitet die Anweisungen, die in der HTML-Seite angegeben sind. Die interpretierte Seite wird nun in dem Browser-Fenster dargestellt.

mAGENTa ist als Java-Applet realisiert. Ein Applet kann als eine Art Programm verstanden werden, welches sich auf einer Internetseite befindet, und innerhalb eines Browser-Fensters abläuft. Um ein Applet korrekt ausführen zu können, muß der Browser „Java-fähig" sein. Das Java-Programm wird über das Netzwerk in einem plattformunabhängigen Bytecode übermittelt und durch den Browser interpretiert.

Da *mAGENTa* mittels der Java 2 Platform entwickelt wurde, muß der Browser über ein JRE (Java Runtime Environment) verfügen, welches Java-Bytecode ab dem JDK (Java Development Kit) 1.2 interpretieren kann.

Zur Zeit der Drucklegung der Diplomarbeit war dies nur dem Browser HotJava von Sun Microsystems möglich, da die Browser der Firmen Netscape und Microsoft bisher nur Java-Applets der Version 1.1.7 ausführen können.

Eine Alternative zu einem vollständigen Internet-Browser ist der AppletViewer der Java 2 Platform von Sun Microsystems. Dieses Programm kann keine kompletten HTML-Seiten darstellen, sondern nur die in sie eingebetteten Java-Applets ausführen. Für die Benutzung von *mAGENTa* reicht dies aber vollkommen aus, da nur in geringem Maße auf HTML-Seiten während der Benutzung von *mAGENTa* zugegriffen werden muß.

Als besonderer Komfort muß *mAGENTa* nicht direkt über ein Netzwerk verwendet werden, sondern kann auch lokal auf einem Arbeitsplatzrechner ausgeführt werden. Dieser ist dann Client und Server zugleich.

Um als Client mit *mAGENTa* zu arbeiten, muß in den Browser die Adresse des Servers und der HTML-Seite, in die *mAGENTa* eingebettet ist, eingegeben werden. Sobald die Seite und das Applet über das Netzwerk geladen worden sind, kann *mAGENTa* in Betrieb genommen werden.

11.1.1.1 Sicherheitsbeschränkungen

Bei der Verbindung eines Computers mit einem Netzwerk, insbesondere dem Internet, ist dieser Computer prinzipiell auch anderen Computern, die an das Netzwerk angeschlossen sind, zugänglich.

Dies könnte nun böswilligerweise dazu genutzt werden, Daten auf diesem Computer zu manipulieren oder zu zerstören. Damit diese Vorgehensweise nicht durch Java-Applets geschieht, die sich ein ahnungsloser Nutzer über ein Netzwerk auf seinen Computer lädt, ist in die Browser eine sogenannte „sandbox“ integriert. Dies bedeutet, daß herkömmliche Java-Applets nur innerhalb eines eng begrenzten Rahmens (der „sandbox“) auf einen Client-Computer zugreifen können. Es ist vor allem nicht möglich, direkt auf Speicherbereiche des Betriebssystems, die Festplatte oder den Drucker zuzugreifen.

Wird das Applet allerdings lokal in dem Appletviewer ausgeführt, so treten die Einschränkungen bzgl. des Lesens und Schreibens auf der Festplatte, sowie des Druckens nicht auf, da hier davon ausgegangen wird, daß lokal ausgeführte Applets grundsätzlich vertrauenswürdig sind und keine Gefahr für den Computer darstellen.

11.1.2 Der Server

Auf dem Server müssen die HTML-Seiten, die Klassen von *mAGENTa*, sowie optionale Zusatzdateien in verschiedenen Verzeichnissen abgelegt werden, damit diese von einem Client angefordert werden können. Das Ablegen der Dateien auf dem Server kann entweder durch direktes Einspielen der Dateien vor Ort oder mittels eines FTP (File Transfer Protocol) - Programmes geschehen.

Als Server reicht ein einfacher HTTP (Hypertext Transfer Protocol) - Server aus, auf dessen angebotene Seiten ein Client lesend zugreifen kann.

11.1.2.1 Die HTML-Seite

Die HTML-Seite, in die das Applet eingebettet ist, sollte minimal den folgenden Eintrag für das Applet beinhalten:

HTML-Tag für das Applet
APPLET ARCHIVE="Magenta.jar" CODE="ProjektApplet.class" WIDTH="130" HEIGHT="80" ALIGN="MIDDLE" ALT="Ihr Browser muß mindestens die Java-Version 1.2 unterstützen"> </APPLET>

Tabelle 13 HTML-Tag für das Applet

Wie zu sehen ist, gibt es hier insgesamt sechs Parameter, die der Browser zur Darstellung des Applets benötigt.

`ARCHIVE="Magenta.jar"`

Das ist der Verweis auf die JAR (Java ARchive) - Datei, in der sich die *mAGENTa*-Klassen befinden. Eine JAR-Datei ist grob betrachtet eine komprimierte Datei, in der sich weitere Dateien, insbesondere Klassendateien befinden. Das hat den Vorteil, daß neben der geringeren Größe nur ein HTTP-Request benötigt wird, um die benötigten Klassendateien (bei *mAGENTa* insgesamt 106) zu übertragen. Dadurch wird die Geschwindigkeit erhöht und die Netzlast beträchtlich vermindert.

Die Datei `Magenta.jar` muß sich in demselben Verzeichnis wie die HTML-Seite befinden, da der Browser sie sonst nicht findet; es sein denn, es ist eine relative oder absolute Adressierung auf die JAR-Datei angegeben.

`CODE="ProjektApplet.class"`

Dies ist die Referenz auf die Applet-Klasse von *mAGENTa*. Dem Browser wird hier mitgeteilt, daß dies die Klasse ist, die zuerst aufgerufen werden soll. Da vorher der `ARCHIVE`-Parameter angegeben wurde, muß sich diese Datei in der JAR-Datei `Magenta.jar` befinden. Diese Klasse ruft direkt oder indirekt alle anderen Klassen von *mAGENTa* auf.

`WIDTH="130"`

`HEIGHT="80"`

`ALIGN="MIDDLE"`

Diese drei Parameter dienen zur Formatierung des Applets innerhalb des Browser-Fensters. Sie geben an, daß der Platz, der dem Applet zur Verfügung steht, 130 Pixel breit, 80 Pixel hoch sowie horizontal in der Mitte der Seite ausgerichtet ist.

```
ALT="Ihr Browser muß mindestens die Java-Version 1.2 unterstützen"
```

Der Parameter `ALT` dient nur dazu eine Fehlermeldung in dem Browser anzugeben, wenn das Applet nicht ausgeführt werden kann.

11.1.2.2 Die Verzeichnisstruktur

Für eine sinnvolle Anwendung reicht es normalerweise nicht aus, nur das Programm zur Verfügung zu stellen, sondern es sollten auch weitere Daten zur Verfügung gestellt werden, die es dem Benutzer erlauben produktiv mit *mAGENTa* zu arbeiten. Hier ist insbesondere an vorgenerierte Graphen, anonymisierte Befunde, erstellte Wörterbücher aus verschiedenen Domänen, personalisierte Briefköpfe und -füße, sowie individuelle Voreinstellungen der Farbgebung der Graphen zu denken.

Damit der Client dies nutzen kann, sollten die entsprechenden Dateien in den folgenden Verzeichnissen auf dem Server zu finden sein:

Verzeichnisname	Dateiendung
Graphen	.graph
Befunde	.txt
Dictionaries	.dic
Briefteile	.txt
Farbschemata	.cs

Tabelle 14 Verzeichnisse auf dem Server

Diese Verzeichnisse sollten sich sinnvollerweise in dem selben Verzeichnis wie die HTML-Datei und die JAR-Datei befinden. Die Namen der Dateien in ihnen müssen die angegebenen Endungen besitzen, damit sie bei der Dateiauswahl auch korrekt angezeigt werden können.

11.2 Inbetriebnahme

Nachdem die HTML-Seite erstellt worden ist, und diese mit den übrigen o.a. Dateien auf dem Server abgelegt worden sind, kann der Client über das Netzwerk die HTML-Seite des Servers aufrufen. Sobald diese komplett geladen worden ist, startet der Browser automatisch das *mAGENTa*-Applet.

Eine komplette Demo-Version von *mAGENTa* mit über 50 Graphen und 400 Befunden ist unter folgender Adresse zu finden:

http://homepage.ruhr-uni-bochum.de/Kai.Bullerdick/Diplomarbeit/index.html

Auf diesem Server sind darüber hinaus weitere Dokumente, wie Zusammenfassungen der Arbeit und Folien zu einem Vortrag, über das *mAGENTa*-System zur Einsicht vorhanden.

11.3 Warum die Java 2-Platform?

mAGENTa wurde mit der Zielsetzung entworfen, dem Benutzer ein Werkzeug an die Hand zu geben, welches ihn bei seiner Arbeit unterstützt und nicht behindert. Dieser Prämisse folgend wurde die Benutzeroberfläche entwickelt.

Für herkömmliche Softwaresysteme, welche in Netzwerken eingesetzt werden, stellt die Heterogenität der Netzwerke durch die verschiedenartigen verbundenen Systeme und Subnetze (besonders ausgeprägt im Internet) eine besondere Schwierigkeit dar. Es findet sich kaum noch ein Netzwerk, welches nur noch Computer mit identischer Hardware und gleichem Betriebssystemen (also ausschließlich Windows 98 oder ausschließlich Solaris) verbindet. Selbst die Art des Netzwerkes kann lokal unterschiedlich sein (z. B. verschiedene Ethernet-Varianten, LocalTalk, ATM,...).

Trotzdem ist es wichtig, daß ein System mit diesen verschiedenartigen Begebenheiten umgehen kann. Mit der Verbreitung des TCP/IP-Standards auf verschiedenen Hardwareplattformen ist zumindest das Problem der Datenübertragung in heterogenen Netzwerken gelöst. Darüber hinaus ist es aber bisher notwendig gewesen, für jedes Betriebssystem, auf dem eine Netzwerkanwendung laufen sollte, eine eigene Lösung zu implementieren, da z. B. für jedes Betriebssystem eine eigene Benutzeroberfläche erstellt werden muß.

Von Anfang an war es ein Bestreben der Entwickler der Programmiersprache Java, eine plattformunabhängige Entwicklungsumgebung zur Verfügung zu stellen, mit der getreu dem Motto "write once, run anywhere" Anwendungen betriebssystemunabhängig erstellt werden konnten. Da insbesondere die Oberflächen-API für jedes Betriebssystem separat erstellt werden mußte, führte dies bisher nicht zu den gewünschten Ergebnissen, sondern vielmehr dazu, daß ein einmal in Java geschriebenes Programm zwar auf jedem Betriebssystem lief, aber durch betriebssystemeigene Aufrufe doch jeweils unterschiedliches Aussehen hatte. Abgesehen davon, daß sich der Benutzer für jedes Betriebssystem an eine neue Oberfläche mit evtl. abweichender Funktionalität gewöhnen mußte, konnten auch die Entwickler bei der Implementierung nicht sicher sein, daß ein einmal entworfenes Programm auf allen Plattformen das gleiche Aussehen hatte.

Seit den Java Foundation Classes (JFC) [Meyer, 1998] ist auch dieses Hindernis für echte plattformunabhängige Software genommen. Als herausragende Erweiterung der JFC des JDK 1.1 ist Swing zu nennen. Seit der Java 2-Platform ist Swing fester Bestandteil der Programmierschnittstelle (API). Mittels Swing können die schwergewichtigen („heavyweight“) Komponenten, die durch das Betriebssystem

dargestellt und verwaltet werden, durch leichtgewichtige („lightweight“) Komponenten ersetzt werden, die selbst komplett durch Java erstellt und verwaltet werden.

Dadurch, daß die Swing-Komponenten plattformunabhängig sind, ist es sogar möglich, zur Laufzeit eines Java-Programmes dessen Layout zu einem von dem Benutzer gewünschten Layout zu verändern. Ein Benutzer, der z.B. die Windows-Oberfläche gewöhnt ist, kann diese mit einem Klick einstellen; der Motif-Gewöhnte kann mit einem Klick seine Wunsch-Oberfläche wieder herstellen (vgl. Kapitel 11.5.2.4.3.4.6 "Look & Feel"). Voreingestellt ist jedoch die Swing-Oberfläche „Metal“, die einen einheitlichen Standard zur Bedienung von Java-Programmen bietet. Bei den Bildschirmdarstellungen der folgenden Kapitel wird dieses „Look & Feel“ benutzt, um sie nicht mit redundanten Darstellungen zu überfrachten.

11.4 Globale Oberflächenkonzepte

Ein modernes Softwaresystem sollte nicht nur funktional, sondern auch ergonomisch sein. Das System sollte leicht zu erlernen sein, damit nicht viel Zeit bei aufwendigen Schulungen verloren geht. Außerdem sollte es einfach zu bedienen sein, damit während der Anwendung nicht viel Zeit mit der Bedienung des Systems verbraucht wird. Diese Zeit kann sinnvoller in der Lösung der Aufgabe, für die das System unterstützend wirkt, genutzt werden. Im Falle eines Befundungsunterstützungssystems wie *mAGENTa* heißt dies, daß der Arzt weniger Zeit damit verbringt, das System zu bedienen, und dadurch mehr Zeit für eine ausführliche Befundung zu Verfügung hat.

Nach der Erklärung herausragender Konzepte für die Oberflächengestaltung von *mAGENTa* folgen Abbildungen, die diese Konzepte anschaulich darstellen.

11.4.1 Adaptivität / Kontext-Sensitivität

Die Adaptivität von *mAGENTa* spiegelt sich nicht nur im TokenComponent-Interface wieder, sondern auch in der Benutzeroberfläche der übrigen Teile von *mAGENTa*. Falls eine Funktion zu einem Zeitpunkt nicht ausführbar ist, so ist der Knopf oder Menüpunkt, über den diese Funktion normalerweise aufgerufen wird, auch nicht anwählbar und optisch dementsprechend dargestellt.

Wenn *mAGENTa* z. B. gestartet wird, so ist es noch nicht sofort möglich, einen DFA abzuspeichern, da bisher noch keiner eingelesen oder generiert worden ist. Erst nachdem der Benutzer einen Befund eingegeben oder einen DFA eingelesen hat, kann dieser abgespeichert werden. Der Benutzer hat somit jederzeit einen Überblick über die Möglichkeiten zur weiteren Vorgehensweise. Hierdurch erfolgt eine zusätzliche Anleitung des Benutzers durch die Funktionalität des Systems.

11.4.2 Automatische Synchronisation

Die Fenster von *mAGENTa* stehen alle in einem inneren Zusammenhang, der auf den DFA als Repräsentation der gelernten Daten zurückgeht. Der Graph und der MetaGraph

zeigen eine direkte graphische Ansicht, das TokenComponent-Interface eine Abbildung dessen und das Wörterbuch die einzelnen Wörter der Zustandsübergangsfunktion.

Eine Änderung des DFA z. B. durch das Lernen von neuen Texten muß also eine direkte Auswirkung auf das Erscheinungsbild der übrigen Fenster haben. Werden z. B. in eine EmptyComponent des TCI oder das Wörterbuch neue Einträge vorgenommen, wirkt sich das wiederum auch auf den DFA und dessen graphische Repräsentation aus.

Damit dies gewährleistet werden kann, herrscht in *mAGENTa* das Prinzip der automatischen Synchronisation vor. Eine Änderung in einem Fenster, einem Dialog oder Menü zur Einstellung von Parametern zieht die sofortige Änderung der Inhalte der anderen Fenster nach sich. Wird zu einem schon gelernten DFA ein neuer Text gelernt, so kann direkt beobachtet werden, wie sich der neue Text auf das Lernergebnis auswirkt.

Da es sich bei dem Lernprozeß bei entsprechend großen Textmengen um eine länger andauernde Aufgabe handeln kann, ist es selektiv auch möglich, die automatische Synchronisation zu verzögern. Wird z.B. in dem Dialog *Festlegen der Merging-Parameter* (siehe Kapitel 11.5.2.4.3.1.1 "Festlegen der Merging-Parameter...") die Option *Sofortige Anwendung der Regeln* aktiviert, so wird sowohl bei Änderung der Merging-Regeln, als auch direkt nach Eingabe in eine EmptyComponent im TCI sowie bei Änderung der Wörterbucheinträge ein neuer Lernvorgang begonnen.

Ist das Auswahlfeld nicht selektiert, so wird in dem Fenster des TCI und des Wörterbuches der *Merge*-Knopf aktivierbar, falls dort eine Änderung vorgenommen worden ist. Erst nachdem dieser Knopf betätigt worden ist, wird ein neuer Lernprozeß gestartet. Insbesondere bei multiplen Eintragungen oder Änderungen in das Wörterbuch ist es sinnvoll, nach Abschluß aller Änderungen den Lernprozeß anzustoßen, anstatt ihn automatisch nach jeder Änderung beginnen zu lassen.

11.4.3 Umfangreiche Unterstützung von Maus- und Tastatureingabe

Der ungeübte Benutzer kann *mAGENTa* (bis auf die Eingabe von Befundtexten in der Lernphase) fast vollständig mit der Maus bedienen. Dem fortgeschrittenen Benutzer steht zusätzlich die Möglichkeit offen, Befehle mittels der Tastatur einzugeben. Bei Systemen, die über keine Maus verfügen, ist dies die einzige Eingabemöglichkeit.

Jeder anklickbaren Oberflächenkomponente ist ein "Shortcut" zugeordnet. Mittels einer plattformspezifischen Meta-Taste auf der Tastatur (z.B. bei Windows die ALT-Taste) und eines passenden weiteren Tastendruckes wird der entsprechende Befehl durchgeführt.

Der Shortcut für eine Komponente läßt sich daran erkennen, daß ein Buchstabe in der Beschriftung der Komponente unterstrichen dargestellt wird. Der unterstrichene Buchstabe muß dann zur Ausführung der gewünschten Operation zusammen mit der Meta-Taste gedrückt werden.

Zwischen einzelnen Oberflächenkomponenten kann auch mittels der Tabulator-Taste manövriert werden. Ist eine Komponente selektiert (erkennbar an dem Rahmen um die Beschriftung), so kann mittels einmaligem Druck auf die Tabulator-Taste die nächste Komponente selektiert werden; bei gleichzeitigem Niederdrücken der Shift- und der Tabulator-Taste wird die vorhergehende Komponente ausgewählt. Diese Vorgehensweise wird auch Fokustraversierung genannt.

11.4.4 Hilfetexte

Den wichtigsten Oberflächenkomponenten ist darüber hinaus noch ein kurzer Hilfetext zugeordnet. Dieser Hilfetext wird sichtbar, sobald der Mauszeiger eine Sekunde über einer solchen Komponente verharrt. Der Text gibt Hinweise auf die Funktion dieser Komponente, falls dem Benutzer trotz der mnemonischen Beschriftung die Benutzung unklar sein sollte.

Zusätzlich existiert das Benutzerhandbuch auch als HTML-Dokument zur Hilfestellung während der Benutzung von *mAGENTa* (siehe auch Kapitel 11.5.2.4.4 "Das Hilfe-Menü").

11.4.5 Anpaßbare Fenster

Für besonderen Benutzerkomfort wurde für *mAGENTa* konsequent die Fenstertechnik benutzt. Jeder Hauptbereich von *mAGENTa* wird in einem eigenen Fenster (auch Frame genannt) dargestellt.

Die Fenster
Hauptfenster
TokenComponent-Interface
Wörterbuch
Graph Ansicht
MetaGraph Ansicht
Befundfenster

Tabelle 15 Die Fenster von mAGENTa

Diese Fenster haben die hinlänglich bekannte Funktionalität gewöhnlicher Betriebssystemfenster wie:

Fensterfunktionalität
Verschieben
Größe verändern
Minimieren
Maximieren
Schließen

Tabelle 16 Fensterfunktionalität

Die Fenster können natürlich nebeneinander auf dem Bildschirm dargestellt werden. Dies ermöglicht es dem Benutzer die für ihn relevanten Informationen aus den einzelnen Fenstern direkt im Blickfeld zu haben. Für größtmöglichen Informationsgehalt kann aber auch ein Fenster auf Bildschirmgröße maximiert werden. In Fenstern, in denen mehr dargestellt werden soll, als es die Größe des Fensters zur Zeit erlaubt, werden jeweils am rechten und ggf. unteren Rand des Fensters Rollbalken dargestellt, damit der Benutzer trotzdem den weiteren Inhalt des Fenster einsehen kann.

Den Hauptfenstern ist gemein, daß sie über eine Werkzeugleiste und ein Menü (nur Hauptfenster und Wörterbuch) verfügen. Die Werkzeugleiste beinhaltet Knöpfe, die bestimmte Operationen auslösen können. Der Einsatz einer Werkzeugleiste ermöglicht den direkten Zugriff auf häufig genutzte Operationen, ohne daß der Benutzer jedesmal durch die Menüstruktur manövrieren muß.

Diese beiden Oberflächenkomponenten werden normalerweise am oberen Rand eines Fensters dargestellt. Die Werkzeugleisten von *mAGENTa* können jedoch auf Wunsch horizontal oder vertikal am Rand des Ursprungsfensters oder gar frei schwebend auf der Oberfläche positioniert werden. Damit eine Werkzeugleiste bewegt werden kann, muß der Mauszeiger auf dem gepunkteten Bereich der Werkzeugleiste (im Falle des Metal-„Look & Feel“) positioniert werden, und mittels der betriebssystemabhängigen Vorgehensweise für „Drag & Drop“ (bei Windows: linke Maustaste gedrückt halten und ziehen) an die gewünschte Position gebracht werden.

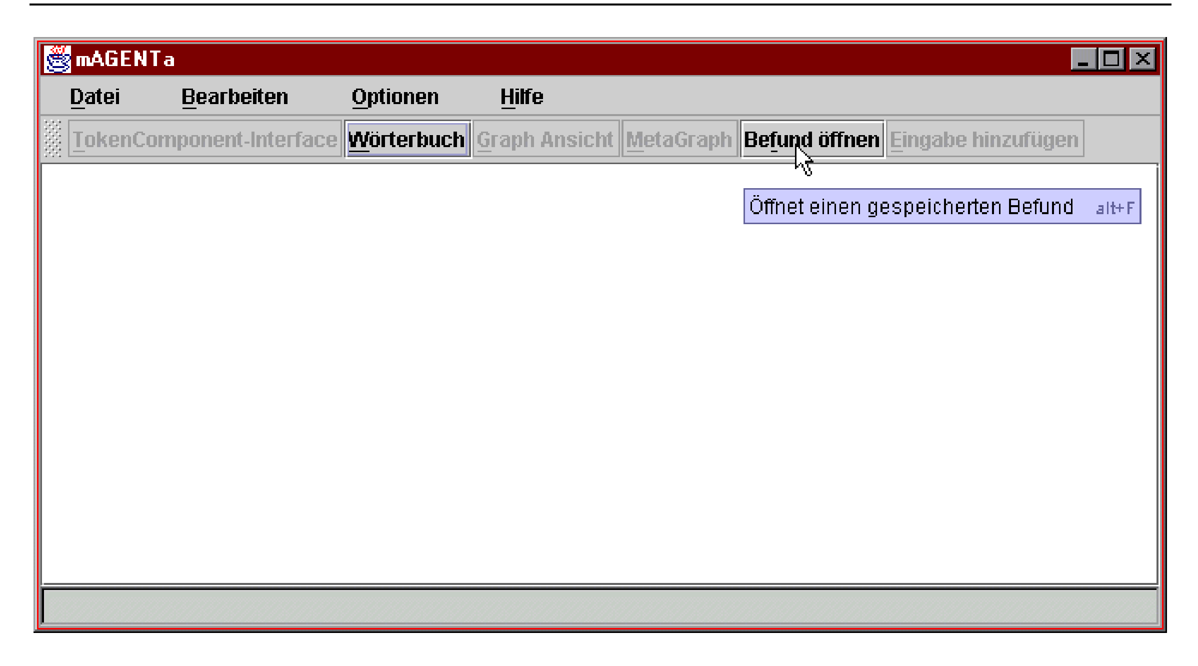


Abbildung 47 Oberflächenkonzepte I

Beispielhaft werden in der obigen Abbildung die gerade aufgezählten Oberflächenkonzepte anschaulich dargestellt. Zu sehen ist das Hauptfenster von *mAGENTa* unmittelbar nach dem Start (weiteres hierzu in Kapitel 11.5.2 "Das Hauptfenster").

Von den sichtbaren Knöpfen sind nur die Knöpfe *Wörterbuch* und *Befund öffnen* aktivierbar. Der Mauszeiger befindet sich über letztgenanntem Knopf. Daraufhin erscheint der Hilfetext *Öffnet gespeicherten Befund alt+F*. Der Zusatz *alt+F* bedeutet, daß in dieser Betriebssystemumgebung der Knopf auch durch die Tastenkombination "Alt" und "F" aktiviert werden kann. Die Tastenkombination ist auch daran zu erkennen, daß das "f" in *Befund öffnen* unterstrichen ist.

Sobald die Voraussetzungen für die Aktivierung einer anderen Oberflächenkomponente vorliegen, wird diese auch als anwählbar dargestellt.

In Abbildung 47 sind das Menü und die Werkzeugleiste am oberen Rand des Fensters positioniert.

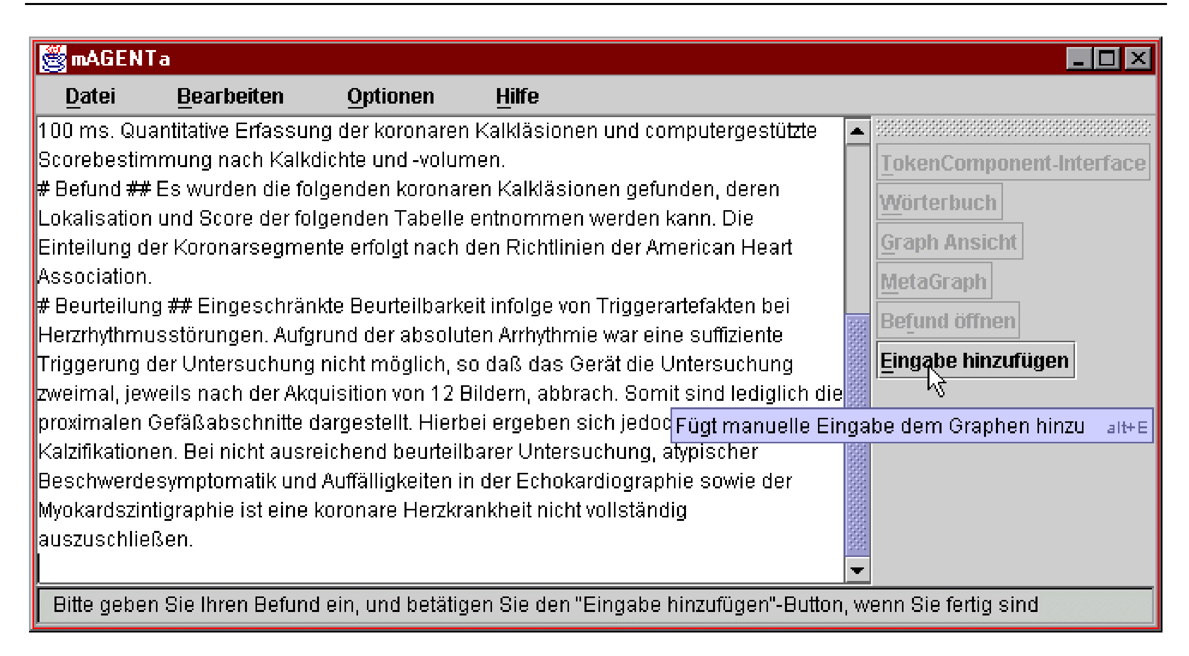


Abbildung 48 Oberflächenkonzepte II

Die obige Abbildung zeigt *mAGENTa*, nachdem ein Befund über die Tastatur eingegeben worden ist.

Die Werkzeugleiste ist hier an der rechten Seite verankert. Die darauf befindlichen Knöpfen sind bis auf den Knopf *Eingabe hinzufügen* nicht anwählbar, da der Benutzer erst mit seiner Eingabe abschließen muß, bevor andere Fenster mit aktualisiertem Inhalt dargestellt werden können.

Da der eingegebene Befund größer ist, als es der sichtbare Bereich zuläßt, wurde automatisch am rechten Rand des Textes ein Rollbalken eingefügt, mit dem der Benutzer den gesamten Text einsehen kann.

11.5 Die Oberfläche

Dieses Kapitel beschreibt die Fenster und Dialoge von *mAGENTa*. Es kann einerseits als Einführung in die umfangreiche Funktionalität gesehen werden, ohne daß der Benutzer die theoretischen Vorkenntnisse aus den vorangegangenen Kapiteln vorweisen muß. Hierfür ist dieses Kapitel so aufgebaut, daß es versucht, den Leser in der Reihenfolge durch die einzelnen Abschnitte zu führen, in der ein Benutzer während der Anwendung von *mAGENTa* die einzelnen Fenster aufrufen würde. Andererseits verfügt dieses Kapitel auch über eine klare Gliederung, die es als Nachschlagewerk während der Benutzung von *mAGENTa* prädestiniert. Die Verweise in einzelnen Abschnitten auf andere Kapitel ermöglichen es die spezifischen Lesewünsche zu unterstützen und evtl. auftretende Fragen gezielter zu beantworten.

Beginnend mit der Ansicht nach Aufruf der HTML-Seite, in die das *mAGENTa*-Applet eingebettet ist, werden die einzelnen Fenster (vgl. Tabelle 15) beschrieben. Zuerst das

Hauptfenster, von wo aus die übrigen Fenster aufgerufen werden können. Anschließend das Fenster mit der Graph-Ansicht, gefolgt von dem ähnlich aufgebauten Fenster mit der MetaGraph-Ansicht, dem Wörterbuch-Fenster, dem Fenster für das TokenComponent-Interface und dem Fenster mit dem fertiggestellten Befund.

In den Abschnitten für die einzelnen Fenster werden zuerst einleitende Worte die grundlegende Aufgabe dieses Fensters erklären. Es folgt eine Erläuterung der Werkzeugleiste und, falls vorhanden, eine Beschreibung des Menüs des Fensters. Eventuell von diesem Fenster aus aufrufbaren Dialoge werden ebenfalls erklärt.

11.5.1 Start des Applets

Nachdem die HTML-Seite, wie in Kapitel 11.2 "Inbetriebnahme" beschrieben, aufgerufen wurde, kann das Browser-Fenster mit dem *mAGENTa*-Applet beispielsweise wie folgt aussehen:

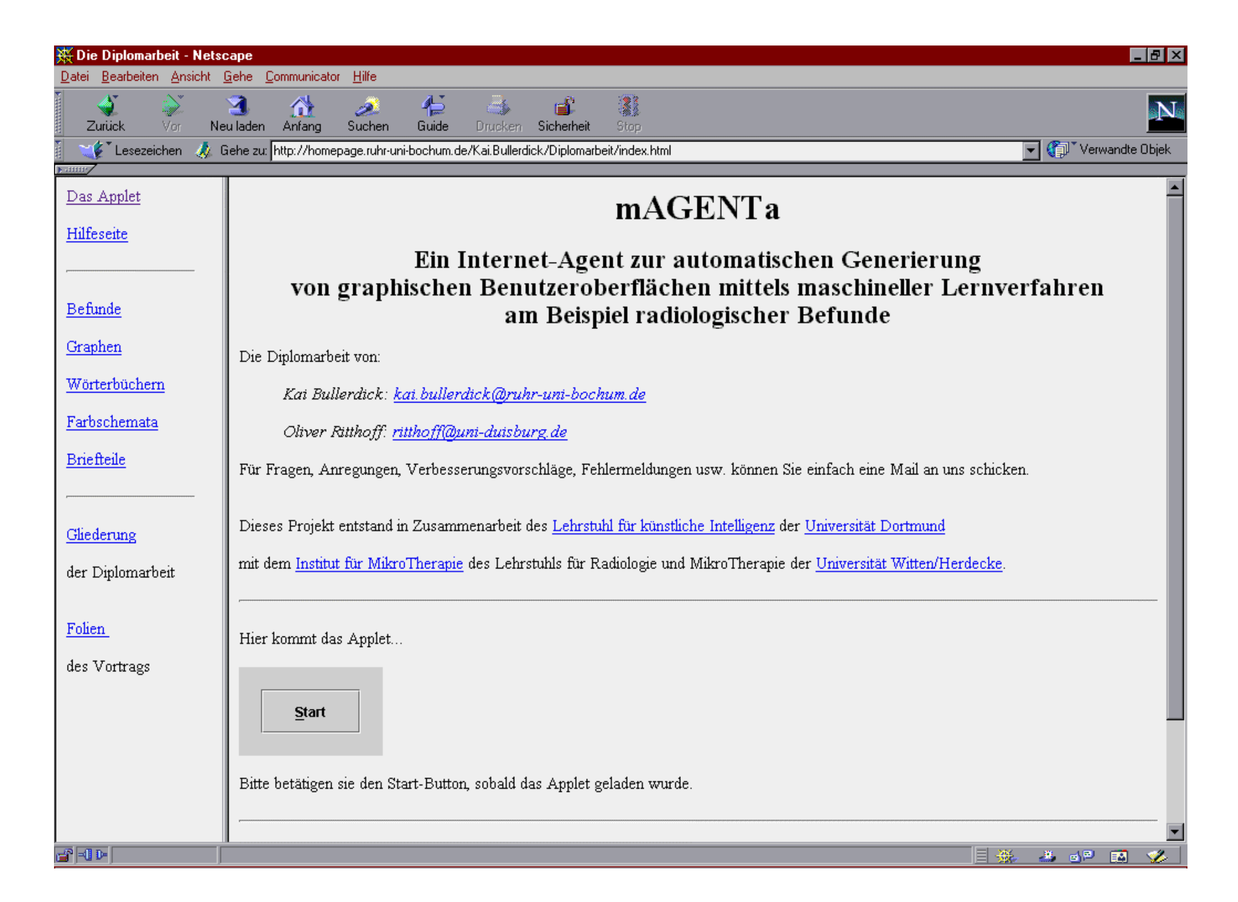


Abbildung 49 Applet Start

Zu sehen ist ein Browser der Firma Netscape mit einer Beispiel HTML-Seite, in die das Applet eingebettet ist. Das Applet wird als graues Rechteck in dem größeren der beiden HTML-Rahmen dargestellt.

In dem Applet selbst ist nur ein einzelner Knopf mit der Beschriftung *Start* sichtbar. Sobald dieser Knopf betätigt wird, öffnet sich das Hauptfenster von *mAGENTa* (siehe Kapitel 11.5.2 "Das Hauptfenster").

Der Knopf dient zur Initialisierung des Applets. Sobald er gedrückt wird, werden die Parameter von *mAGENTa* auf die voreingestellten Anfangswerte gesetzt. Dies ist insbesondere auch während der Laufzeit möglich, damit der Benutzer eventuelle Änderungen an der Parametrisierung wieder rückgängig machen kann. Zusätzlich werden alle bis dahin geöffneten Fenster geschlossen. Anschließend wird das Hauptfenster jedoch wieder geöffnet. Damit auch optisch zu erkennen ist, daß es sich um eine Neuinitialisierung handelt, ändert sich die Beschriftung des Buttons nach Betätigung von *Start* auf *Neustart*.

Die Voreinstellungen finden sich in dem Anhang Voreinstellungen.

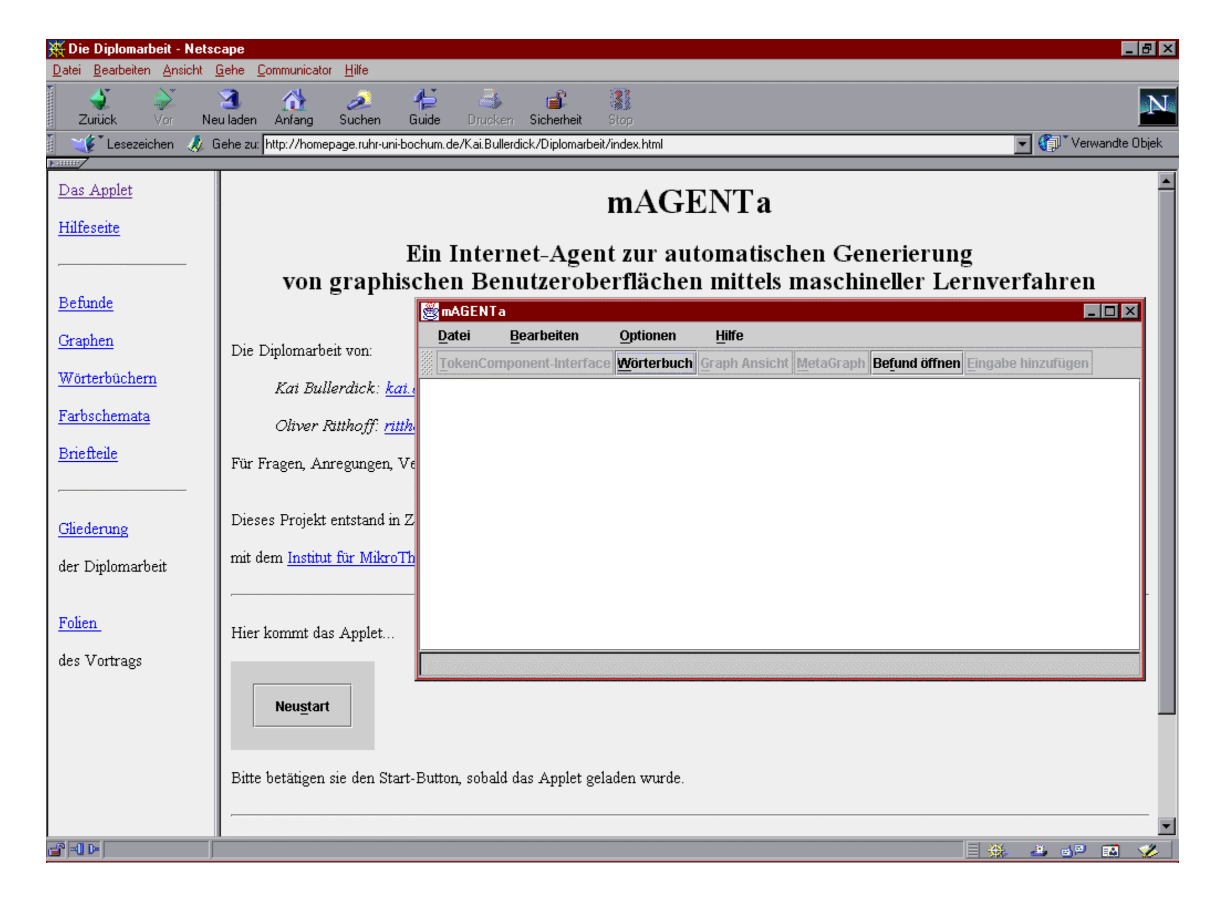


Abbildung 50 Applet Neustart

11.5.2 Das Hauptfenster

Das Hauptfenster ist sozusagen die Schaltzentrale von *mAGENTa*. Dies ist Ausgangspunkt und Zentrum des Systems, von hier aus können die anderen Fenster aktiviert, Dateien geöffnet und gespeichert, sowie die Dialoge zur Einstellung der umfangreichen Parameter aufgerufen werden.

Die folgende Abbildung zeigt das Hauptfenster von *mAGENTa* direkt nach dem Start. Am oberen Rand befinden sich das Menü und darunter die Werkzeugleiste. Der Großteil des Fensters ist mit dem Textfeld für die manuelle Befunderfassung belegt. Am unteren Rand ist die Statuszeile sichtbar.

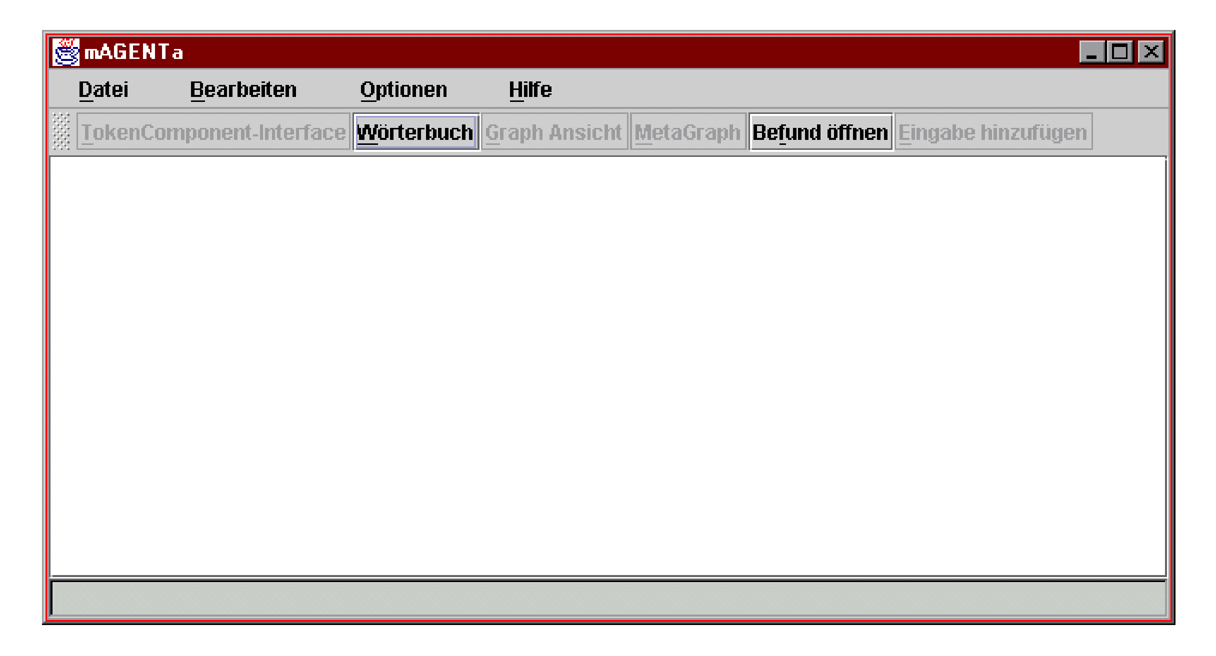


Abbildung 51 Hauptfenster

11.5.2.1 Das Textfeld für die Befundeingabe

In dieses Feld kann der Benutzer direkt seine Texte eingeben, die von dem System gelernt werden sollen.

Sobald der Benutzer mit der Maus einmal in das Textfeld geklickt hat oder mittels der Tabulator-Taste den Selektions-Fokus auf das Textfeld wandern ließ, kann dort der zu lernende Text eingegeben werden.

Wie bei allen Textfeldern in *mAGENTa* ist hier ein automatischer Zeilenumbruch vorhanden. Das heißt, sobald der eingegebene Text länger als die vorhandene Zeilenlänge wird, so wird die Zeile umgebrochen, und der Text wird in der nächsten Zeile weitergeführt. Dies führt allerdings nicht zu unerwünschten Trennungen eines einzelnen Wortes, sondern das Wort, welches nicht mehr in die obere Zeile paßt, wird an den Anfang der nächsten Zeile übertragen.

Sollte der Text über mehr Zeilen reichen, als ursprünglich in dem Textfeld zur Verfügung standen, so wird automatisch am rechten Rand des Textfeldes ein Rollbalken erscheinen, um den Text zeilenweise nach oben zu rollen. Mit Hilfe des Rollbalkens kann der Benutzer die nicht mehr in dem sichtbaren Bereich des Textfeldes befindlichen Zeilen einsehen, indem er ihn nach oben verschiebt.

Der Beginn der Texteingabe bewirkt, daß die Knöpfe der Werkzeugleiste bis auf den Knopf *Eingabe hinzufügen* deaktiviert werden. Erst wenn mittels dieses Knopfes die Eingabe als vollständig bestätigt wird, beginnt der Lernprozeß mit den eingestellten Parametern (vgl. Kapitel 11.5.2.4.3.1.1 "Festlegen der Merging-Parameter"). Wenn dieser abgeschlossen ist, sind die übrigen Knöpfe aktivierbar und der *Eingabe hinzufügen*-Knopf wird wieder deaktiviert, bis eine erneute Eingabe in das Textfeld erfolgt.

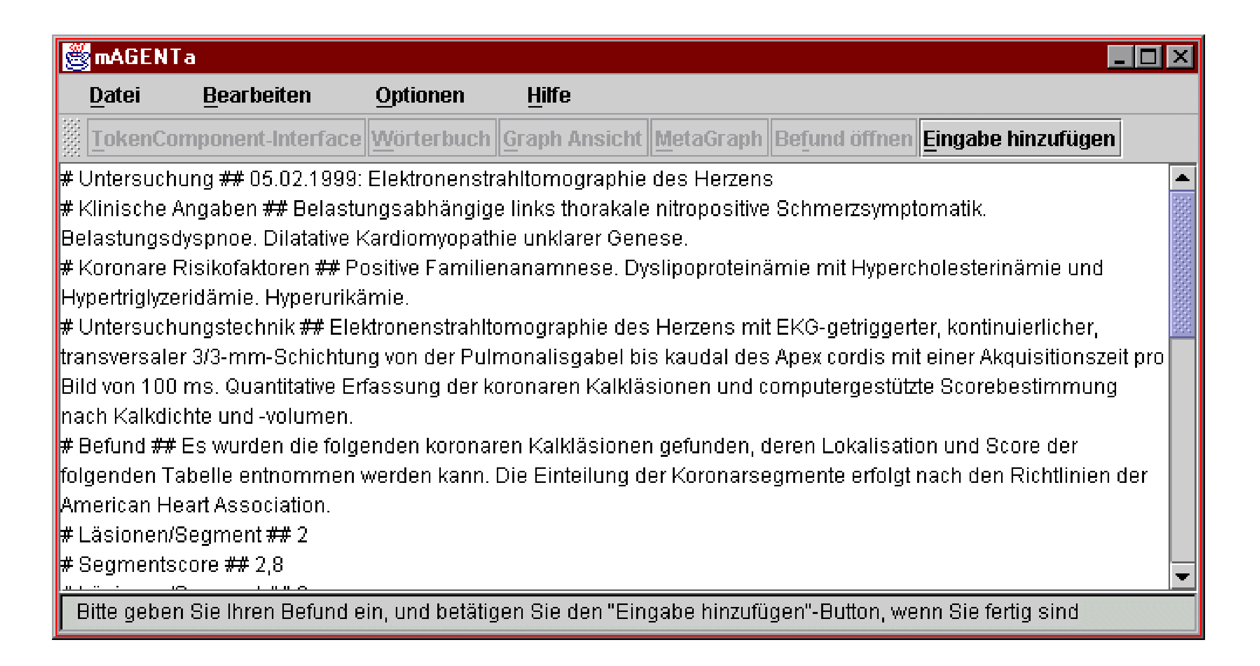


Abbildung 52 Eingabe in das Hauptfenster

Die obige Abbildung zeigt das Hauptfenster bei einem manuell eingegebenen Befund. Zu bemerken ist hierbei, daß jeder MetaAbschnitt tatsächlich jeweils in nur einer einzigen Zeile liegt, und diese am rechten Rand des Textfeldes entsprechend umgebrochen wurde.

11.5.2.2 Die Statuszeile

In ihr werden für die jeweiligen von dem Benutzer ausgelösten Aktionen kurze Mitteilungen über deren Status angezeigt. Dies geschieht sowohl bei Aktionen, die über die Werkzeugleiste aktiviert werden, als auch über Aufrufe durch das Menü.

Bei vielen Aktionen erhält der Benutzer Hinweise auf die weitere sinnvolle Vorgehensweise. Z. B. wird bei der Eingabe eines Textes in das Textfeld eine Meldung eingeblendet, die dem Benutzer mitteilt, daß ,sobald die Eingabe abgeschlossen ist, der *Eingabe hinzufügen*-Knopf zu drücken ist (vgl. Abbildung 52 Eingabe in das Hauptfenster).

Bei länger andauernden Aktionen wird dort zusätzlich eine Uhr eingeblendet, die Auskunft darüber gibt, wie lange die aktuelle Operation gerade dauert. Zum Beispiel hat der Benutzer bei einem Lernprozeß den temporären Fortschritt des Vorgangs direkt im Blickfeld.

11.5.2.3 Die Werkzeugleiste

Wie aus Abbildung 51 ersichtlich, sind nach dem Start von *mAGENTa* nur die Knöpfe *Wörterbuch* und *Befund öffnen* aktivierbar. Dies folgt dem Konzept der Kontext-Sensitivität (vgl. Kapitel 11.4.1 "Adaptivität / Kontext-Sensitivität"), da es nur mit einem vorhandenen DFA sinnvoll ist, die Fenster, die über die anderen Knöpfe erreichbar sind, zu öffnen. Sobald jedoch ein DFA generiert (siehe Kapitel 11.5.2.1 "Das Textfeld für die Befundeingabe", 11.5.2.3.1 "Eingabe hinzufügen", 11.5.2.3.2 "Befund öffnen", bzw. 11.5.2.4.1.6 "Befundliste einlesen"), oder geladen (siehe Kapitel 11.5.2.4.1.2 "Öffnen", bzw. 11.5.2.4.1.3 "Hinzufügen") worden ist, werden diese jedoch aktivierbar, und der Benutzer kann die dadurch aufgerufenen Fenster nutzen.

Im Folgenden werden die Funktionen der einzelnen Knöpfe erläutert.

11.5.2.3.1 Eingabe hinzufügen

Wie schon in Kapitel 11.5.2.1 "Das Textfeld für die Befundeingabe" erwähnt, ist dieser Knopf zu betätigen, wenn die Eingabe in das Textfeld abgeschlossen werden soll, d.h. wenn ein manuell erfaßter Befund vollständig eingegeben wurde.

Sobald dieser Knopf betätigt worden ist, wird der Lernalgorithmus den neuen Text mit den eingestellten Parametern dem bisher Gelernten hinzufügen. Falls kein Fehler bei dem Lernprozeß auftrat, wird anschließend der Inhalt des Textfeldes gelöscht, um für neue Eingaben bereit zu stehen.

Falls jedoch ein Fehler während des Lernvorganges auftrat, erscheint eine Fehlermeldung, die über die Ursache des Fehlers Auskunft gibt. Der eingegebene Befund bleibt in dem Textfeld erhalten, und der Benutzer kann seinen Befund dahingehend ändern, daß der Text bei dem nächsten Anstoß des Lernverfahrens fehlerlos gelernt werden kann.

Tabelle 17 zeigt die im Zusammenhang mit dem Einlesen eines Befundes potentiell auftretenden Fehler (Exception), zusammen mit der im jeweiligen Fall angezeigten Fehlermeldung und einer kurzen Erläuterung des entsprechenden Fehlers. [10]

Exception	Fehlertext	Beschreibung
BefundLeer-Exception	„Der Befund ist leer“	Der eingelesene Befund enthält keinen Text
KeinAbschnitts-TextException	„Es wurde kein Text für den Meta-Abschnitt angegeben“	Ein Meta-Abschnitt enthält keinen entsprechenden Abschnitts-Text
KeinTitel-Exception	„Es wurde kein MetaTitel angegeben“	Ein als Meta-Zustand markierter Zustand enthält keinen entsprechenden Meta-Titel
KorrekterBeginnException	„Eingabesequenz beginnt nicht mit einer Markierung“	Der eingelesene Text enthält einen End-Delimiter, ohne daß zuvor ein entsprechender Start-Delimiter angegeben wurde
KorrektTerminiertException	„Eine Endekennung wurde nicht angegeben“	Der eingelesene Text enthält zwar einen Start-Delimiter jedoch keinen entsprechenden End-Delimiter

Tabelle 17 potentielle Fehler beim Einlesen eines Befundes

11.5.2.3.2 Befund öffnen

Ein Befund kann natürlich nicht nur über die Tastatur eingegeben werden. Falls er schon in elektronisch lesbarer Form als Datei vorliegt, kann er direkt eingelesen werden.

Der Befund muß hierzu in dem Unicode-Format vorliegen. Unicode ist eine 16-Bit-Zeichenkodierung, im Gegensatz zu der herkömmlichen ASCII- und ISO88591 (Latin-1)-Kodierung, die nur ein Byte benötigt. Der Unicode-Standard enthält 38.885 unterschiedliche Zeichen aus den weltweit 25 wichtigsten Sprachen, bzw. Schriften. Da die 256 Unicode-Zeichen \u0000 bis \u00FF (16-Bit Unicode-Zählweise) mit den Zeichen 0x00 bis 0xFF (8-Bit Hexadezimal-Zählweise) der oben genannten Kodierungen identisch sind [Flanagan, 1998], können auch ganz normale Textdateien eingelesen werden. Auch dies unterstreicht wieder den flexiblen Grundgedanken von *mAGENTa*.

Die Befunddatei kann sich entweder lokal auf der Festplatte befinden oder auch innerhalb des Netzwerkes, auf das der Benutzer Zugriff hat.

[10] Zum besseren Verständnis der nachfolgenden Tabelle sei in diesem Zusammenhang auf das Kapitel 7.1.1 verwiesen.

Je nachdem, ob die Option *Internet-Verbindung vorhanden* (vgl. Kapitel 11.5.2.4.3.4.7 "Internet-Verbindung vorhanden") selektiert ist, wird ein anderer Dateiauswahldialog geöffnet.

Da die Vorgehensweise zum Öffnen von anderen Dateiarten (Graphen : Kapitel 11.5.2.4.1.2 "Öffnen...", Wörterbücher : Kapitel 11.5.5.3.1.2 "Öffnen...", Farbschemata : Kapitel 11.5.2.4.3.1.3 "Farbauswahl...") ähnlich ist, wird hier eine ausführlichere Beschreibung geliefert und in den o.a. Kapiteln hierauf verwiesen.

11.5.2.3.2.1 Öffne lokale Datei

Ist die Option *Internet-Verbindung vorhanden* nicht selektiert, so wird ein Dateiauswahldialog geöffnet, mit dem der Benutzer auf die lokalen Verzeichnisse zugreifen kann.

Der Dialog zeigt zuerst das voreingestellte Verzeichnis, in dem sich die Befunddateien befinden, an (siehe Tabelle 14 Verzeichnisse auf dem Server"). Aus der Tabelle geht auch hervor, über welche Dateiendung die Datei verfügen muß, um in dem Dialog angezeigt zu werden. Die übrigen Dateien, die nicht über die spezifische Dateiendung verfügen, werden der Übersichtlichkeit halber nicht angezeigt.

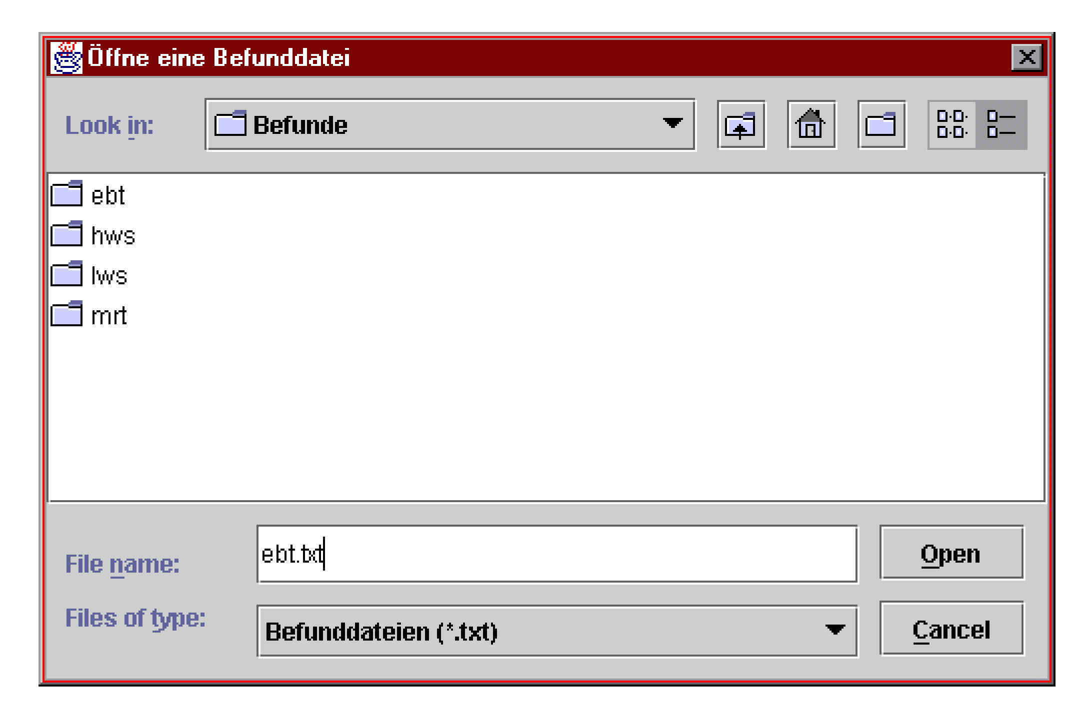


Abbildung 53 Öffne lokale Datei

In der obigen Abbildung ist ein solcher Dateidialog dargestellt. Beachtet werden muß, daß Dateiauswahldialoge stark betriebssystemspezifisch sind und demzufolge in einem anderen eingestellten „Look & Feel“ auch deutlich anders aussehen.

In dem oberen Bereich des Dialoges findet sich eine Auswahlliste, über die direkt die einzelnen Laufwerke und deren Verzeichnisse ausgewählt werden können, ein Knopf zum Wechsel in das über dem aktuellen Verzeichnis liegende Verzeichnis, ein Knopf zum Sprung in das Home-Vezeichnis, ein Knopf zur Erstellung eines neuen Verzeichnisses und ein Knopf zum Wechsel der Ansicht der Dateien (als Listen- oder als Detail-Ansicht).

Darunter befindet sich der Auswahlbereich, in dem die einzelnen Dateien und Verzeichnisse des aktuellen Verzeichnisses aufgelistet werden. Mittels eine Doppelklicks auf ein Datei- oder Verzeichnis-Symbol wird das entsprechende geöffnet. Dies kann auch durch einen einzelnen Klick auf den *Open*-Knopf nach Selektion des gewünschten Elementes geschehen.

Neben diesem Knopf befindet sich ein Feld zur Eingabe des Dateinamens. Darunter ist die Auswahlliste der selektierbaren Dateitypen zu sehen, die durch diesen Dateiauswahldialog geöffnet werden können.

Beendet wird dieser Dialog durch Betätigen des *Open*-Knopfes nach Selektion einer Datei oder durch Betätigen des *Cancel*-Knopfes bzw. das Schließen des Fensters.

Falls eine Datei selektiert worden ist, so wird sie von dem lokalen Laufwerk eingelesen und analog der Vorgehensweise in Kapitel 11.5.2.3.1 "Eingabe hinzufügen" dem bisher Gelernten hinzugefügt.

11.5.2.3.2.2 Öffne externe Datei aus dem Netzwerk

Im Unterschied zum Öffnen einer lokalen Datei kann über den in dem folgenden abgebildeten Dialog eine Datei aus einem heterogenen Netzwerk geöffnet werden.



Abbildung 54 Öffne externe Datei

In diesen Dialog muß nur die URL (Uniform Resource Locator, die Netzwerkadresse) der zu öffnenden Datei eingegeben werden, um sie zu über das Netzwerk zu transferieren und dem Gelernten hinzuzufügen. Bei der Eingabe der URL ist stets auch das verwendete Protokoll zu benennen. In der obigen Abbildung wurde das HyperText Transport Protocol (HTTP) verwendet. Wird allerdings das File-Protocol („file://...“)

und der Name einer Datei auf dem lokalen Dateispeicher angegeben, so kann lesend auch auf diese Datei zugegriffen werden.

Da die URL einer Datei mitunter sehr lang werden kann, stellt *mAGENTa* eine wirkungsvolle Hilfe zur Verfügung.

Abhängig vom Kontext (d.h. welche Dateiart geöffnet werden soll) wird der im Hintergrund laufende Internet-Browser angewiesen, die Indexdateien der passenden Dateien auf dem Server anzuzeigen. Mittels dieser Indexdateien kann der Benutzer auf dem Server zwischen den Verzeichnissen manövrieren. Er kann die gewünschte Datei auswählen und auch in dem Browserfenster ansehen, bevor sie von *mAGENTa* geladen wird.

Die folgende Abbildung illustriert diesen Vorgang:

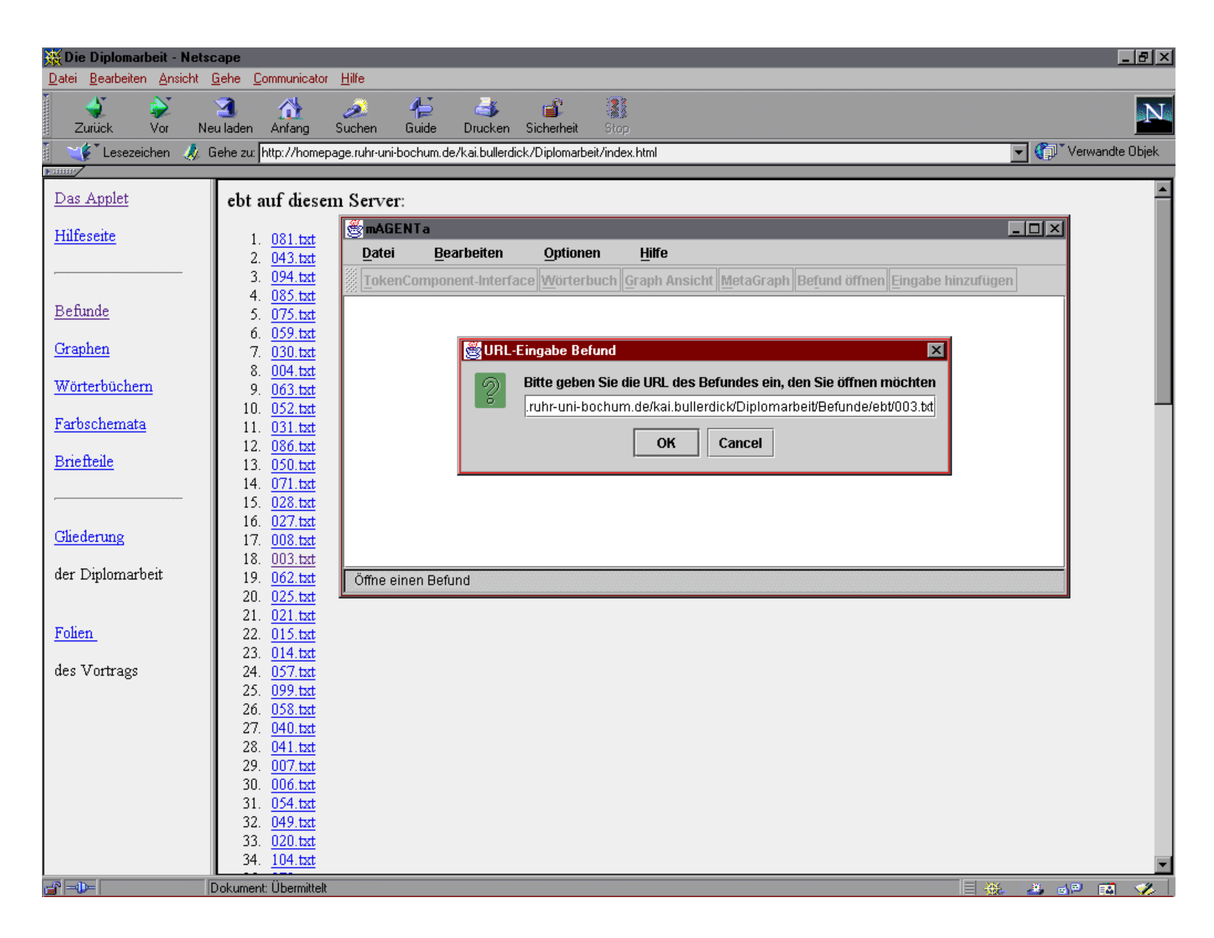


Abbildung 55 Öffne externe Datei mittels eines Internet-Browsers

Wird die gewünschte Datei in dem Browserfenster angezeigt, so kann mittels der rechten (sekundären) Maustaste die Verknüpfungsadresse kopiert werden und anschließend in das Textfeld des Eingabedialoges übertragen werden. Nun kann die Datei wie oben beschrieben über das Netzwerk geladen werden.

Die Indexdateien wurden mit dem Hilfsprogramm *generateIndex* (vgl. Kapitel 11.7 "generateIndex"), welches auch im Rahmen der Diplomarbeit geschrieben wurde, erstellt.

11.5.2.3.3 Graph Ansicht

Dieser Knopf ist nur dann aktivierbar, wenn bisher etwas gelernt worden ist. Die Betätigung dieses Knopfes öffnet das Fenster mit einer anschaulichen, dynamischen graphischen Visualisierung des Graphen. Dies wird ausführlich in Kapitel 11.5.3 "Der Graph" beschrieben.

11.5.2.3.4 MetaGraph

Es öffnet sich das Fenster mit der Visualisierung des MetaGraphen. Dieser Knopf ist allerdings nur aktivierbar, wenn ein Graph existiert, d.h. etwas gelernt worden ist und die Option *MetaStruktur erwünscht* selektiert worden ist (vgl. Kapitel 11.5.2.4.3.4.1 "MetaStruktur erwünscht"). Eine ausführliche Darstellung dieses Fensters ist in Kapitel 11.5.4 Der MetaGraph zu finden.

11.5.2.3.5 Wörterbuch

Mittels dieses Knopfes kann das Fenster des Wörterbuches geöffnet werden. Weiteres hierzu im Kapitel 11.5.5 "Das Wörterbuch".

Im Gegensatz zu den anderen Fenstern ist dieses auch dann zu öffnen, wenn bisher kein Graph gelernt worden ist. Dies liegt an einer gewissen Unabhängigkeit der Wörterbücher von den Graphen. Sie können unabhängig voneinander geöffnet, gespeichert und verändert werden.

11.5.2.3.6 TokenComponent-Interface

Ebenso öffnet sich über diesen Knopf das Fenster des TokenComponent-Interfaces nur, wenn vorher ein Graph gelernt worden ist. Eine ausführliche Erläuterung der TCI-Oberfläche findet sich in Kapitel 11.5.6 "Das TokenComponent-Interface".

11.5.2.4 Die Menüs

Neben der Werkzeugleiste stellen die Menüs eine wichtige Schnittstelle zur Benutzerinteraktion dar. Angelehnt an bewährte und weit verbreitete Standards gibt es auch in *mAGENTa* ein *Datei*-Menü, ein *Bearbeiten*-Menü, ein *Hilfe*-Menü und darüber hinaus ein *Optionen*-Menü.

11.5.2.4.1 Das Datei-Menü

Die Einträge des *Datei*-Menüs des Hauptfensters beziehen sich jeweils auf den gesamten Graphen, bzw. das gesamte System.

11.5.2.4.1.1 Neu

Der Menüpunkt *Neu* veranlaßt *mAGENTa* dazu, das bisher gelernte zu "vergessen". Der Graph wird aus dem Speicher gelöscht, und sämtliche Parameter auf die Voreinstellungen (siehe Anhang Voreinstellungen) zurückgesetzt. Ähnlich verhält es sich bei dem Betätigen des Knopfes *Neustart* (vgl. Kapitel 11.5.1 Start des Applets") mit dem Unterschied, daß nicht alle übrigen Fenster geschlossen werden.

11.5.2.4.1.2 Öffnen...

Mit diesem Menüpunkt kann ein Graph eingelesen werden. Mit dem Unterschied, daß die Graphdatei die Endung "*.graph*" haben muß, gelten die gleichen Ausführungen wie in Kapitel 11.5.2.3.2 "Befund öffnen".

Sollte vorher schon einmal ein Graph geöffnet oder generiert worden sein, so wird er durch den neu eingelesenen ersetzt.

Der Dateiname der aktuell geöffneten Graphen wird anschließend in der Titelzeile des Hauptfensters angezeigt.

In der Statuszeile wird nach dem Öffnen des Graphen zur Information die Dauer des Vorganges, sowie die Anzahl der Zustände und der Tokens angezeigt.

11.5.2.4.1.3 Hinzufügen...

Das *Öffnen* eines Graphen hat "Gedächtnisverlust" zur Folge. Damit *mAGENTa* aber auf den bisherigen Erfahrungen aufbauend weiterlernen kann, gibt es den Menüpunkt *Hinzufügen*.

Mit dem Aufruf dieser Operation wird, wie im vorigen Abschnitt beschrieben ein Graph geladen. Nur wird hier nicht ein evtl. vorhandener Graph durch den neuen ersetzt, sondern der neue wird dem alten unter Anwendung der eingestellten Merging-Parameter (siehe Kapitel 11.5.2.4.3.1.1 "Festlegen der Merging-Parameter...") hinzugefügt. Der alte Graph lernt also von dem neuen.

Ist noch kein Graph vorhanden, so verhält sich *Hinzufügen* wie *Öffnen*.

11.5.2.4.1.4 Speichern

Dieser Menüpunkt speichert den Graphen unter dem bisherigen Dateinamen. Der Dateiname ist *mAGENTa* dadurch bekannt, daß der Graph aus einer Datei geladen worden ist, oder vorher die Datei mittels des Menüpunktes *Speichern unter* (siehe nächsten Abschnitt) unter einem einzugebenden Namen gespeichert worden ist.

Dieser Menüpunkt ist also nur aktivierbar, wenn *mAGENTa* einen Graphen gelernt hat, dieser unter einem Dateinamen bekannt ist und außerdem die Option *Internet-Verbindung vorhanden* nicht selektiert ist. Eine Datei kann aus Sicherheitsgründen nicht auf einem entfernten Rechner abgespeichert werden, sondern dies kann nur auf einem lokalen Laufwerk geschehen.

11.5.2.4.1.5 Speichern unter...

Ein neu generierter Graph kann durch diesen Menüpunkt unter einem einzugebenden Dateinamen lokal abgespeichert werden. Sobald dieser Menüpunkt ausgewählt worden ist, öffnet sich ein Dateiauswahldialog ähnlich dem *Öffnen*-Dialog für eine lokale Datei (vgl. Kapitel 11.5.2.3.2.1 "Öffne lokale Datei").

Der äußere Unterschied besteht nur darin, daß der Knopf, der vorher mit *Open* betitelt war, hier mit *Save* beschriftet ist.

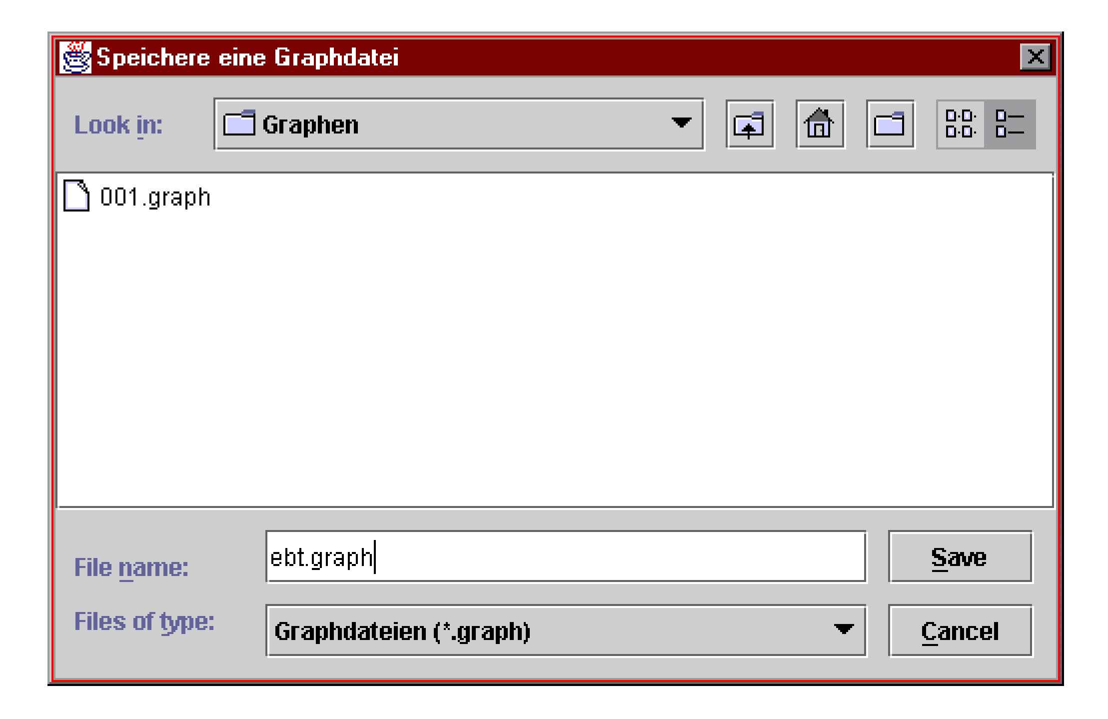


Abbildung 56 Speichere eine Graphdatei

Wie im vorigen Abschnitt schon dargelegt, ist die Speicherung von Dateien außerhalb des lokalen Systems nicht möglich. Sollte dies gewünscht sein, so kann die Datei (die entsprechenden Zugriffsrechte vorausgesetzt) mittels FTP auf den Server zur Dateiablage übertragen werden.

Insbesondere für das spätere Öffnen von Graphdateien und deren Übertragung durch das Netzwerk ist es wichtig, daß nicht durch große Dateien eine hohe Netzlast entsteht. Um dies zu vermeiden, werden die Dateien von *mAGENTa* vor dem Abspeichern automatisch komprimiert. Durch das Verwenden des ZIP-Algorithmus zur Komprimierung werden die Dateigrößen auf ca. 20 % verringert.

Beispielsweise wurde ein Graph mit 8480 Zuständen, 9235 Tokens und 17 MetaZuständen von einer ursprünglichen Dateigröße von 1,6 MegaByte auf nur 352 KiloByte reduziert.

Ist eine Graphdatei mit einem Namen versehen und abgespeichert worden, so erscheint dieser Name künftig in der Titelzeile des Hauptfensters.

11.5.2.4.1.6 Befundliste einlesen...

Oftmals ist es notwendig, mehrere Texte hintereinander zu lernen. Die manuelle Eingabe in das Textfeldes des Hauptfeldes wäre hier recht arbeitsintensiv. Selbst, wenn die Texte in elektronischer Form als Textdateien vorlägen, müßte jede einzelne Datei explizit über den Menüpunkt *Öffnen* hinzugefügt werden. Bei mehreren Dutzend oder hunderten von Texten (wie bei den Testreihen im Rahmen der Diplomarbeit) wird selbst die einfache Bedienung der Maus als zu aufwendig betrachtet.

Hierfür bietet *mAGENTa* die Möglichkeit an, Listen von Befunden zu verarbeiten und von dem System lernen zu lassen. Diese Befundlisten sind einfache Textdateien, deren einzelne Zeilen jeweils den Dateinamen eines Textes enthalten, der sich in dem selben Verzeichnis wie die Befundliste befindet.

Sämtliche Dateien aus der Befundliste werden mit den aktuell eingestellten Parametern gelernt.

Die Befundliste selbst wird mit den bekannten Dateiauswahldialogen für das Öffnen von Dateien entweder lokal oder über das Netzwerk eingelesen.

Eine weitergehende Automatisierung des Einlesens und Lernens von sehr vielen Befunden bietet die ergänzend entwickelte Software *mAGENTa*Script (siehe Kapitel 11.6 "*mAGENTa*Script").

11.5.2.4.1.7 Merging-Protokoll anzeigen...

Es nicht nur wichtig, daß etwas gelernt worden ist, sondern es interessiert aus theoretischer Sicht auch insbesondere, *wie* etwas gelernt worden ist.

Hierfür wird von *mAGENTa* ein Protokoll des Lernvorganges erstellt. Insbesondere finden die Protokolle des Lernprozesses unter unterschiedlicher Parametrisierung Anwendung in Kapitel 12 "Vergleichende Testergebnisse".

Jeder Lernvorgang, unabhängig davon ob er durch Eingabe eines Befundes in das Befundfenster, durch Einlesen eines Befundes, bzw. einer Befundliste aus einer Datei, durch Hinzufügen eines neuen Graphen zum bestehenden Graphen, durch Eingabe in eine EmptyComponent, bzw. durch Ersetzungen von Einträgen im Wörterbuch ausgelöst wurde, wird von *mAGENTa* protokolliert. Dieses Protokoll umfaßt eine Reihe von Statistiken bezüglich des entstandenen Graphen, bzw. MetaGraphen, der Merging-Operationen etc

Das Protokoll wird in zwei unterschiedlichen Formatierungen erstellt. Die erste ist ausführlich und auch für Menschen lesbar, die zweite enthält nur die bei der Protokollierung erhobenen Werte. Die zweite Formatierungsart ist für die elektronische Weiterverarbeitung bestens geeignet, da ein Lerndurchlauf genau eine Zeile umfaßt und

die einzelnen Werte durch Semikola getrennt werden. Dies ermöglicht einen einfachen Import in eine Tabellenkalkulation wie z.B. Excel der Firma Microsoft.

Die Werkzeugleiste des Fensters beinhaltet zwei Auswahlknöpfe, mit denen zwischen den Formatierungen gewechselt werden kann: *Ausführlich* und *Nur Werte*.

Die beiden folgenden Abbildungen zeigen das Erscheinungsbild mit den jeweils unterschiedlichen Formatierungen:

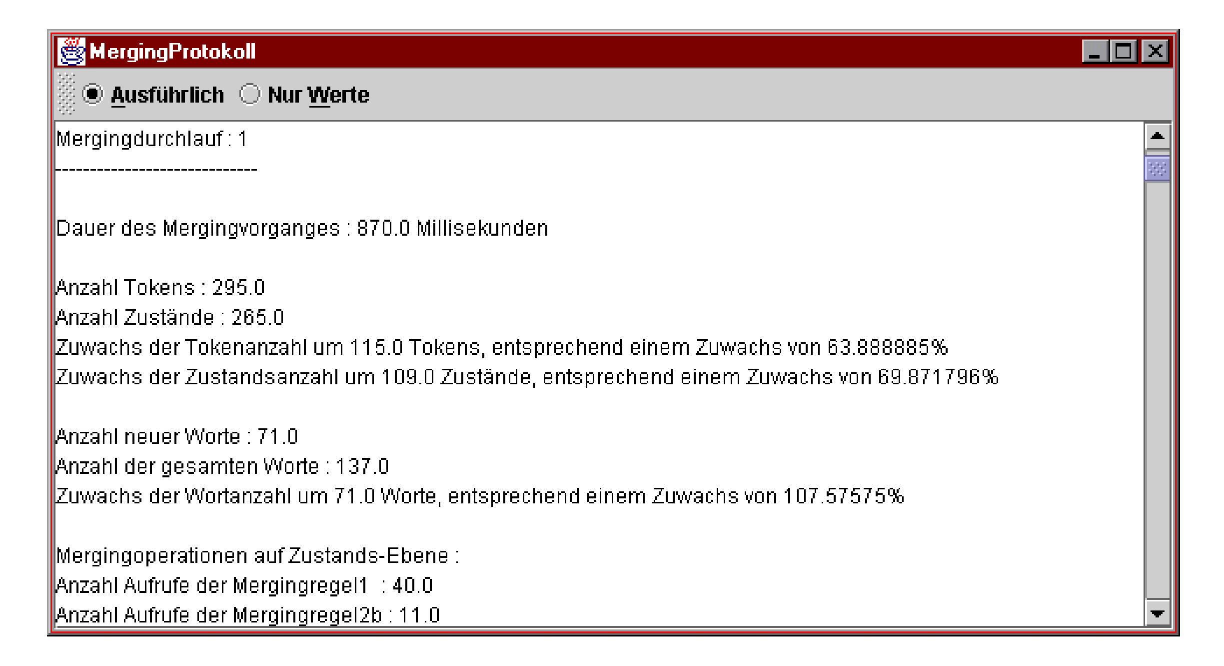


Abbildung 57 MergingProtokoll-Ausführlich

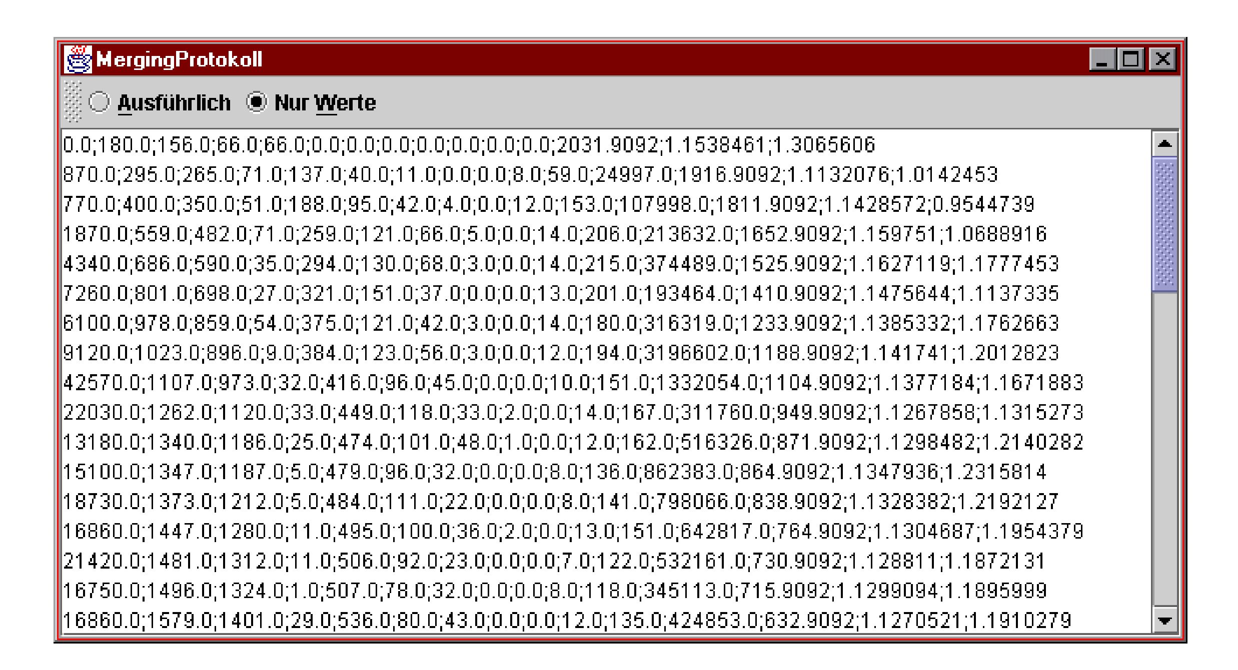


Abbildung 58 MergingProtokoll-Nur Werte

Das Protokoll wird bei jedem Lernprozeß um weitere Daten ergänzt. Die folgende Tabelle gibt Auskunft über die erhobenen Daten:

Einträge des Mergingprotokolls
Fortlaufende Nummer des Lernlaufes
Dauer des Lernvorganges in Millisekunden
Anzahl der jetzt vorhandenen Tokens
Anzahl der jetzt vorhandenen Zustände
Zuwachs der Tokenanzahl: absolut
Zuwachs der Tokenanzahl: prozentual
Zuwachs der Zustandsanzahl: absolut
Zuwachs der Zustandsanzahl: prozentual
Anzahl der jetzt vorhandenen Worte
Zuwachs der Wortanzahl: absolut
Zuwachs der Wortanzahl: prozentual
Anzahl der Aufrufe der Mergingregel 1
Anzahl der Aufrufe der Mergingregel 2b
Anzahl der Aufrufe der Mergingregel 3a
Anzahl der Aufrufe der Mergingregel 3b
Anzahl der Mergingoperationen auf Meta-Ebene
Anzahl gesamten Mergingaufrufe
Anzahl der gesamten Testaufrufe
Prozentualer Anteil der Mergingoperationen an den Testaufrufen
Lineare Abweichung der aktuellen von der mittleren Tokenanzahl
Mittelwert der Anzahl der Zustandsnachfolger
Standardabweichung der Anzahl der Zustandsnachfolger

Tabelle 18 Einträge des Mergingprotokolls

11.5.2.4.1.8 Eigenschaften des Graphen anzeigen...

Einen schnellen Überblick über den Umfang und besondere Charakteristika des Graphen bietet dieser Menüpunkt. In dem sich öffnenden Dialog werden die wichtigsten Daten aufgeführt.

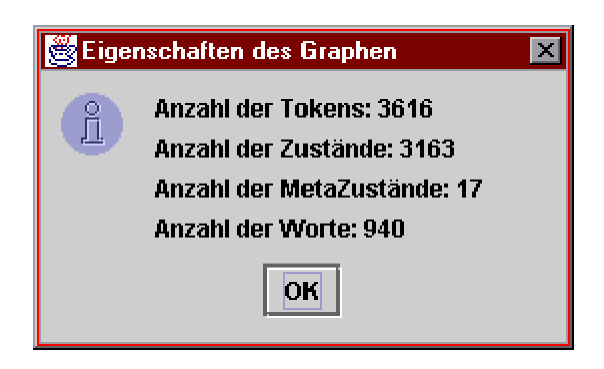


Abbildung 59 Eigenschaften des Graphen

Der Dialog wird durch einen Klick auf den Knopf *OK* wieder geschlossen.

Eine ausführlicheren Blick auf das Lernergebnis liefert der Menüpunkt *Merging-Protokoll* anzeigen... aus dem vorigen Abschnitt.

11.5.2.4.1.9 Beenden

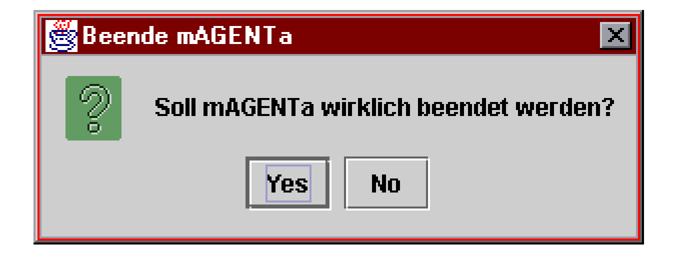


Abbildung 60 Beenden von mAGENTa

Nach Betätigung des Menüpunktes *Beenden* öffnet sich der oben abgebildete Dialog zur Sicherheitsabfrage. Wird diese durch anklicken des Knopfes *Yes* bestätigt, so wird das System zurückgesetzt und sämtliche zur Zeit geöffneten Fenster von *mAGENTa* geschlossen. Das Zurücksetzen bewirkt eine Reinitialisierung der Voreinstellungen (siehe Anhang Voreinstellungen) gefolgt von einer Speicherbereinigung (Garbage-Collection). Wird der Knopf *No* betätigt, so wird der Dialog ohne Änderungen geschlossen.

11.5.2.4.2 Das Bearbeiten-Menü

Die Punkte in diesem Menü dienen der Vereinfachung der Texteingabe. Wie ein Textverarbeitungsprogramm bietet auch *mAGENTa* die Möglichkeit der einfachen Textmanipulation durch Unterstützung der Systemzwischenablage (auch Clip-board genannt). Die Zwischenablage ist ein Speicherbereich, in dem Daten zwischengespeichert werden können, um z.B. von einer Anwendung in eine andere oder auch innerhalb einer einzelnen Anwendung transferiert zu werden.

Bei *mAGENTa* bezieht sich dieser Datentransfermechanismus auf Texteingaben. Es kann also ein Text oder auch einzelne Wörter bzw. Buchstaben aus einer Textverarbeitung importiert oder exportiert werden. Auch ist der Transfer von Text innerhalb einer Eingabe möglich.

In diesem Menü finden sich die drei klassischen Operationen für Daten der Zwischenablage:

Die Zwischenablage-Operationen
Ausschneiden (Cut)
Kopieren (Copy)
Einfügen (Paste)

Tabelle 19 Die Zwischenablage-Operationen

11.5.2.4.2.1 Ausschneiden

Der vorher mit der Maus oder der Tastatur selektierte Text wird durch diese Operation in die Zwischenablage kopiert und anschließend aus dem gesamten Text gelöscht.

11.5.2.4.2.2 Kopieren

Diese Operation arbeitet wie *Ausschneiden* mit dem Unterschied, daß der selektierte Text erhalten bleibt.

11.5.2.4.2.3 Einfügen

Wurde ein Text durch die Operationen *Ausschneiden* oder *Kopieren* in die Zwischenablage gebracht (auch aus anderen Anwendungen), so kann dieser Text durch die *Einfügen*-Operation in das Textfeld (siehe 11.5.2.1 Das Textfeld für die Befundeingabe) eingefügt werden. Die Einfügung findet immer an der Position der Schreibmarke statt. Sollte bisher noch kein Text in das Textfeld eingegeben worden sein und somit keine Schreibmarke sichtbar sein, so erscheint der eingefügte Text am Anfang des Feldes.

Die oben benannten Operationen des Menüs sind auch betriebssystemabhängig über Tastatur-Shortcuts durchführbar (z.B. für Windows-Betriebssysteme: Strg+X für *Ausschneiden*, Strg+C für *Kopieren* und Strg+V für *Einfügen*).

11.5.2.4.3 Das Optionen-Menü

Das Optionen-Menü stellt umfangreiche Konfigurationsmöglichkeiten für *mAGENTa* zur Verfügung. Hier kann eingestellt werden, wie sich das System im weiteren Verlauf verhalten soll (z.B. Einstellungen für den Lernprozeß), außerdem kann hier Einfluß auf

das Erscheinungsbild von einzelnen Fensterinhalten genommen werden (z.B. TokenComponent-Interface).

Die Optionen wurden auf solche Werte eingestellt, daß von Benutzerseite ein subjektiv gutes Ergebnis erzielt wurde. Die voreingestellten Werte sind im Anhang aufgelistet.

Das Optionen-Menü verfügt über drei Untermenüs für die Fenster Graph, TokenComponent-Interface und Wörterbuch, sowie sieben weitere Menüpunkte zur Einstellung von weiteren Parametern.

Einige Menüpunkte sind einfache Auswahlfelder (Checkboxes), andere öffnen Dialoge zur Eingabe der Voreinstellungen.

11.5.2.4.3.1 Die Graph-Optionen

11.5.2.4.3.1.1 Festlegen der Merging-Parameter...

In dem in Abbildung 61 gezeigten MergingDialog können die wichtigsten Parameter des Lernverfahrens eingestellt werden.

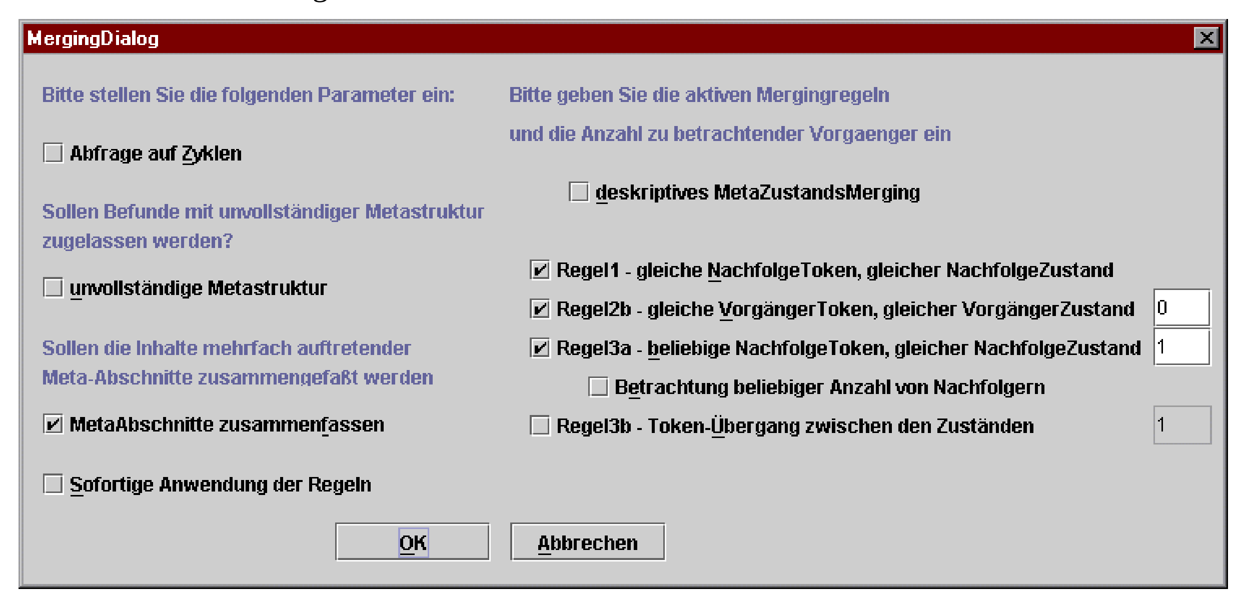


Abbildung 61 MergingDialog

Zunächst kann der Benutzer beim Merging von Meta-Zuständen zwischen dem präskriptiven und dem deskriptiven Ansatz auswählen.[11] Um das deskriptive Verfahren auszuwählen, muß er das Auswahlfeld *deskriptives MetaZustandsMerging* selektieren.

Die beiden nachfolgenden Parameter erlauben eine gewisse Flexibilisierung bei der Festlegung der MetaStruktur der einzulesenden Befundtexte.

[11] Das Konzept der Meta-Struktur und die beiden Merging-Verfahren auf Meta-Ebene werden im Detail in Kapitel 8.3.1, bzw. 8.3.2 vorgestellt.

So kann der Benutzer bestimmen, daß beim Einlesen der Befundtexte auch Befunde mit einer unvollständigen MetaStruktur akzeptiert werden. (siehe Kapitel 8.3.2.4). Diese Option kann durch Aktivierung des Auswahlfeldes *unvollständige MetaStruktur* ausgewählt werden.

Desweiteren kann festgelegt werden, daß MetaAbschnitte mit identischen MetaTiteln innerhalb eines Befundes zusammengefaßt werden sollen. (siehe Kapitel 8.3.2.5) Durch Anwahl des Feldes *MetaAbschnitte zusammenfassen* kann diese Option aktiviert werden.

Tabelle 20 zeigt drei potentielle Fehler im Zusammenhang mit dem Merging von MetaZuständen, die jeweiligen Fehlerausgaben sowie eine kurze Erläuterung der Fehler.

Exception	Fehlertext	Beschreibung
Mehrfache-MetaZustaende-Exception	“Der neue Befund enthält folgende MetaTitel mehrfach :”	Der eingelesene Befund enthält mindestens einen MetaTitel mehrfach und die Option “MetaAbschnitte zusammenfassen” ist nicht aktiviert
Zusätzliche-MetaZustaende-Exception	“Der neue Befund enthält folgende MetaTitel zusätzlich :”	Der eingelesene Befund enthält zusätzlich zu den bereits vorhandenen Meta-Zuständen mindestens einen weiteren Meta-Zustand
Fehlende-MetaZustaende-Exception	“Der neue Befund enthält folgende MetaTitel nicht :”	Beim eingelesenen Befund ist bezogen auf die aktuell vorhandenen MetaZuständen mindestens ein MetaZustand nicht aufgeführt und die Option “unvollständige Metastruktur” ist nicht aktiviert

Tabelle 20 Potentielle Fehler beim Merging von MetaZuständen

Die MergingRegeln auf Zustandsebene, also konkret die in Kapitel 8.3.3.1 beschriebenen Regeln 1, 2a, 2b, 3a, 3aE und 3b können in dem Dialog nahezu beliebig kombiniert werden. So sind durch Aktivierung der entsprechenden Auswahlfelder z. B. auch die Regelmengen von Angluin (1, 2a, 2b) bzw. Schlimmer und Hermens (1, 2a, 2b, 3a, 3b) leicht nachzubilden. Über bestimmte Textfelder kann der Benutzer dabei die Anzahl der k-Vorgänger (siehe Definitionen in Kapitel 8.1) bezüglich der Regeln 2a, 2b, 3a, 3aE und 3b einstellen. Bei der Kombination der Regelmenge hat er die Wahl zwischen der ursprünglichen Regel 3a von Schlimmer und Hermens und der erweiterten Version 3aE.

Des weiteren kann er festlegen, ob ein Lernvorgang auf Grundlage der oben beschriebenen Parameter sofort auf den aktuellen DFA angewendet werden soll, oder *mAGENTa* diesen Vorgang erst nach einer entsprechenden Aktion (Einlesen eines Befundes, Hinzufügen eines Graphen, etc.) auslöst. Eine direkte Anwendung der Mergingparameter kann durch Aktivierung des Auswahlfeldes *Sofortige Anwendung der Regeln* eingestellt werden.

11.5.2.4.3.1.2 Angestrebte Kantenlänge...

Diese Einstellung wirkt sich auf das Verhalten des Graphen bei der Darstellung im Graph-Fenster (siehe Kapitel 11.5.3 "Der Graph") aus.



Abbildung 62 Angestrebte Kantenlänge

Bei der fortlaufenden Aktualisierung der Graph-Ansicht versuchen die einzelnen Knotenpunkte jeweils einen bestimmten Abstand zueinander zu erreichen, um so einen gleichmäßige Verteilung der gesamten Knoten innerhalb des Fensters zu erreichen. Die Länge der Kanten zwischen den Knoten kann hier eingestellt werden. Voreingestellt ist ein Wert von 5 Pixeln. Ein höherer Wert bewirkt, daß die Knoten einen größeren Abstand erhalten (sinnvoll bei einer geringen Anzahl von Knoten und/oder hohen Bildschirmauflösung), ein kleinerer Wert sorgt dafür, daß die Knoten näher zusammen liegen (sinnvoll bei einer höheren Anzahl von Knoten und/oder niedrigen Bildschirmauflösung).

11.5.2.4.3.1.3 Farbauswahl...

Die Farbauswahl-Option dient dazu, die Erscheinung des Graphen und des MetaGraphen farblich zu verändern. Den einzelnen graphischen Elementen lassen sich unterschiedliche Farben zur besseren Identifikation zuordnen. Die Gesamtheit aller Farbeinstellungen, das Farbschema, läßt sich zur späteren Verwendung auch abspeichern. Dadurch wird ermöglicht, daß jeder Benutzer sein eigenes individuelles Farbschema benutzen kann.

Die färbbaren graphischen Elemente	
Zustand	MetaZustand
TerminalZustand	Token
Selektierter Knoten	Fixierter Knoten
Kante von einem Zustand	Kante von einem Token
Kantenlänge	Text

Tabelle 21 Die färbbaren graphischen Elemente

In der obigen Tabelle sind die graphischen Elemente aufgelistet, denen jeweils eine Farbe zugeordnet werden kann. Die Farbe eines Elementes wird jeweils durch ihre Rot-, Grün- und Blau-Komponenten charakterisiert (RGB-Modell). Die einzelnen Komponentenanteile sind ganze Zahlen zwischen 0 und 255. Die Standardfarbe für ein Zustandsknoten beispielsweise hat einen roten Anteil von 255, einen grünen von 201 und einen blauen von 2. Die Voreinstellung für die übrigen Farbwerte findet sich in Anhang Voreinstellungen.

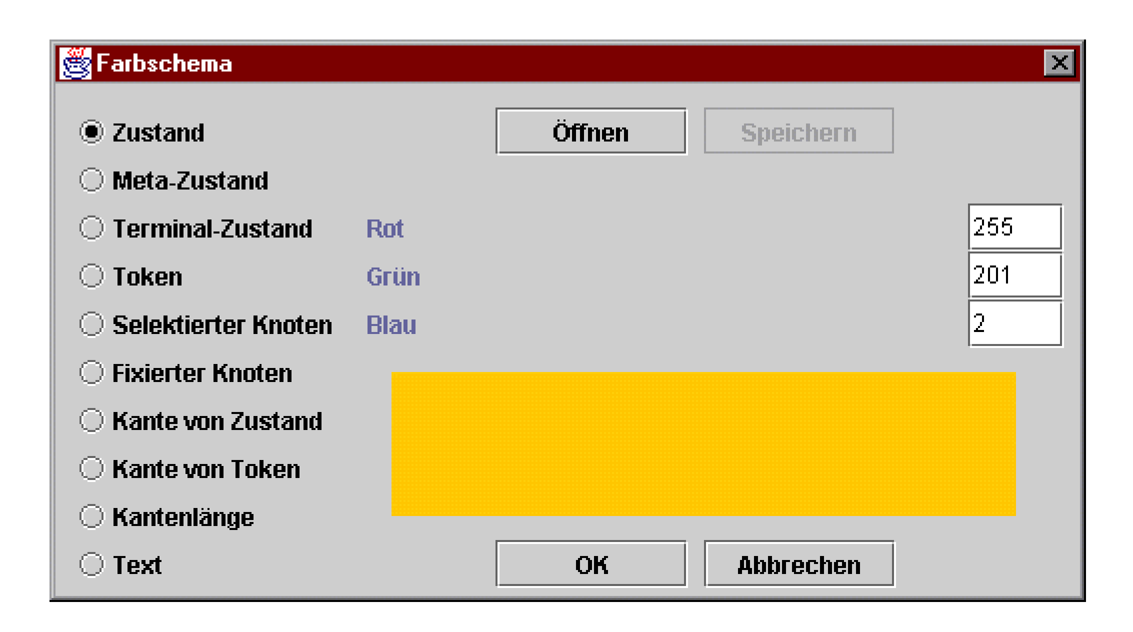


Abbildung 63 Farbauswahldialog

Die Auswahl eines Elementes geschieht über die auf der linken Seite des Dialoges aufgelisteten Auswahlknöpfe. Wird ein Auswahlknopf angeklickt, so erscheinen die Farbkomponentenwerte in den entsprechenden Textfeldern, die Schieberegler werden auf die passende relative Position (zwischen 0 und 255) gesetzt, und das Farbfeld erhält die aktuell gültige Farbe des Elementes.

Die einzelnen Werte für ein Element können entweder über einen Schieberegler oder direkt in ein Textfeld eingegeben werden. Ein Ziehen des Schiebereglers erhöht oder erniedrigt direkt den Wert in dem nebenstehenden Textfeld.

Zur Kontrolle des Farbergebnisses nach Änderung eines Wertes wird in dem Dialog ein rechteckiger Canvas-Bereich entsprechend der resultierenden Farbe eingefärbt. Ist das Graph- oder MetaGraph-Fenster geöffnet, so werden die einzelnen Komponenten parallel zum Ziehen des Schiebereglers eingefärbt. Somit kann der Benutzer direkt die Farbänderungen am aktuellen Graphen während der Aktualisierung beobachten.

Erst wenn der Benutzer die Auswahl seiner Farben beendet hat und den Dialog mittels des *OK*-Knopfes schließt, wird die Änderung der Farben aber in den Graphen bzw.

MetaGraphen übernommen. Wird der Dialog mittels des *Abbrechen*-Knopfes verlassen, so stellt *mAGENTa* die vorher gültige Farbgebung wieder her.

Da nicht jeder Benutzer einen ähnlichen Farbgeschmack hat, kann mittels der Knöpfe im oberen Bereich des Dialoges ein geeignetes Farbschema, wie in Kapitel 11.5.2.4.1.2 "Öffnen..." bzw. Kapitel 11.5.2.4.1.5 "Speichern unter..." für Graphen beschrieben, geöffnet und gespeichert werden.

11.5.2.4.3.2 Die Wörterbuch-Optionen

Für das Wörterbuch kann nur eine Einstellung geändert werden. Weitergehende Informationen bezüglich der Bedienung des Wörterbuches sind in Kapitel 11.5.5 "Das Wörterbuch" zu finden. Eine Beschreibung des selbst Wörterbuches liefert das Kapitel 7.2 "Das Wörterbuch".

11.5.2.4.3.2.1 Maximale Anzahl von Einträgen pro Seite...

Da das Wörterbuch-Fenster über ein eigenes Optionen-Menü mit einem identischen Menüpunkt verfügt wird dieses ausführlich dort beschrieben (siehe Kapitel 11.5.5.3.2.1 "Maximale Anzahl von Einträgen pro Seite...").

11.5.2.4.3.3 Die TokenComponent-Interface-Optionen

Da diese Optionen schon in Kapitel 9 "Das TokenComponent-Interface" ausführlich beschrieben worden sind, erfolgt hier nur noch einmal der Vollständigkeit halber eine kurze Auflistung der Menüeinträge, die eher auf die Bedienung als auf die dahinter liegende Theorie eingeht.

Eine ausführliche Beschreibung der Bedienung des TCI findet sich in Kapitel 11.5.6 "Das TokenComponent-Interface".

11.5.2.4.3.3.1 Anzeige Nachfolger-Symbol

Ist dieses Auswahlfeld selektiert, so werden im TCI Informationen über denjenigen Zustand angezeigt, der erreicht wird, wenn eine TokenComponent betätigt wird.

Die Erklärung für die einzelnen Symbole läßt sich auch im TCI über den Knopf *Legende* aus der Werkzeugleiste aufrufen (siehe Kapitel 11.5.6.4.1 "Legende").

11.5.2.4.3.3.2 Anzeige Steuerzeichen

Sollte dieses Auswahlfeld selektiert sein, werden im TCI die Steuerzeichen (auch Escape-Sequenzen genannt) in der Standardnotation der C-Escape-Sequenzen dargestellt.

Wird der Wert des Auswahlfeldes (selektiert oder nicht selektiert) verändert, und ist gleichzeitig das TCI-Fenster geöffnet, so werden automatisch die Beschriftungen der TokenComponents mit den Steuerzeichen versehen oder diese entfernt. Dies folgt dem

Prinzip der automatischen Synchronisation (siehe Kapitel 11.4.2 "Automatische Synchronisation").

11.5.2.4.3.3.3 Mehrfaches Besuchen von Metazuständen

Ein MetaZustand kann aufgrund eines Zyklus mehrfach besucht werden. Ist dieses nicht gewünscht, d.h. es sollen MetaZustände nur genau einmal besucht werden können, so ist diese Option zu deselektieren. Wird unter dieser Konfiguration bei Eingabe in das TCI ein MetaZustand ein zweites Mal besucht, so wird automatisch der nächste noch nicht besuchte MetaZustand aufgesucht.

Ist diese Option jedoch selektiert, so kann ein MetaAbschnitt, dessen MetaZustand sich in einem Zyklus befindet, beliebig oft durchlaufen werden.

11.5.2.4.3.3.4 Mehrfache Anzeige von Metatiteln

Ist die vorige Option selektiert, so wird bei jedem Besuch des MetaZustandes im TCI der Titel des MetaZustandes in den Befund übernommen, um den Beginn eines neuen MetaAbschnittes zu signalisieren.

11.5.2.4.3.3.5 Nach Frequenz

Das TCI präsentiert dem Benutzer die einzelnen TokenComponents in der Reihenfolge, die durch die Frequenzen der repräsentierten Tokens vorgegeben wird.

Sollte eine Sortierung nach der Frequenz der Tokens nicht gewünscht sein, so ist dieses Auswahlfeld zu deselektieren. Die TokenComponents bleiben an ihrer aktuellen Position verankert, sofern kein neuer Lernprozeß gestartet wird.

11.5.2.4.3.3.6 Protokoll der TC-Positionen

Mit der Selektion dieses Auswahlfeld wird festgelegt, daß die Position einer angeklickten TokenComponent protokolliert werden soll. Am Ende eines Erfassungsvorganges wird eine Auswertung darüber angezeigt.

Ausführliche Informationen über das Protokoll sind in Kapitel 11.5.6.5 "Das Protokoll-Fenster" zu finden.

11.5.2.4.3.3.7 Anzahl TokenChoice...

In dem nach Anwahl des Menüpunktes sich öffnenden Dialoges kann festgelegt werden, wie viele TokenButtons mit gleichem Start- und Endzustand zu einer TokenChoice zusammengefaßt werden sollen.

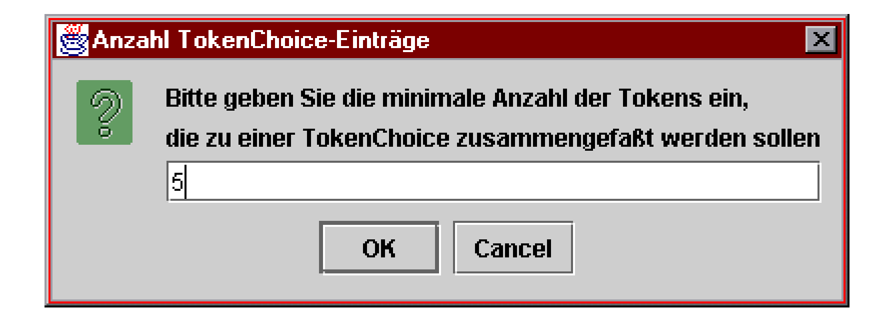


Abbildung 64 Anzahl TokenChoice

11.5.2.4.3.3.8 Briefkopf...

Abhängig davon, ob eine Verbindung zum Internet besteht (siehe Kapitel 11.5.2.4.3.4.7 "Internet-Verbindung vorhanden"), erscheint der entsprechende Dateiauswahldialog (siehe Kapitel 11.5.2.3.2.1 "Öffne lokale Datei", bzw. 11.5.2.3.2.2 "Öffne externe Datei aus dem Netzwerk") zum Öffnen einer Datei für den Briefkopf. Diese sollte eine einfache Textdatei sein, die als Einleitung für den erfaßten Text zu sehen ist.

11.5.2.4.3.3.9 Brieffuß...

Die Ausführungen zu dem vorigen Abschnitt über den Menüpunkt *Briefkopf...* gelten auch für diesen Menüpunkt, mit der Einschränkung, daß hier eine Datei angegeben werden kann, die hinter dem erfaßten Text steht.

11.5.2.4.3.4 Die sonstigen Optionen

11.5.2.4.3.4.1 MetaStruktur erwünscht

Soll bei dem Lernen von Texten deren Struktur gemäß dem MetaKonzept berücksichtigt werden, so ist diese Option zu selektieren.

Ist sie nicht selektiert, so werden die Texte jeweils als Ganzes betrachtet, und das Lernverfahren arbeitet über die Abschnittsgrenzen hinaus.

Die Einstellungen aus dem Kapitel 11.5.2.4.3.1.1 "Festlegen der Merging-Parameter...", die sich auf das Mergen von MetaZuständen beziehen, haben demnach keine Auswirkungen auf den Lernprozeß. Ebenso bleiben die in den Kapiteln 11.5.2.4.3.3.3 "Mehrfaches Besuchen von Metazuständen" und 11.5.2.4.3.3.4 "Mehrfache Anzeige von Metatiteln" vorgestellten Optionen ohne Wirkung, da es keine MetaZustände gibt.

Diese Option kann nur mit einem bisher leeren DFA eingestellt werden. Sobald ein Text gelernt worden ist, kann diese Option nicht mehr verändert werden. Sie ist nicht mehr selektierbar, bis der Benutzer den Menüpunkt *Neu* (siehe Kapitel 11.5.2.4.1.1 "Neu") anwählt.

11.5.2.4.3.4.2 Sortierung der Tokens nach Frequenz

Wie in Kapitel 11.5.2.4.3.3.5 "Nach Frequenz" dargelegt worden ist, verfügen die Tokens über eine Frequenz, die angibt, wie oft sie bisher in gelernten Texten auftraten bzw. wie oft sie bei der Texterfassung im TCI ausgewählt worden sind. Dieser Menüpunkt gibt an, ob die Tokens, die von einem Zustand in einen anderen überführen, gemäß ihrer Frequenz sortiert werden sollen. Dies wirkt sich dann auf das TCI aus, da die TokenComponents von Tokens mit höherer Frequenz vor denen mit niedrigerer dargestellt werden.

Bei dem Lernprozeß ist festzustellen, daß die Sortierung der Tokens sich positiv auf die Geschwindigkeit des Lernvorganges auswirkt. Tokens, die bisher häufig in Texten vorkamen, haben eine hohe Wahrscheinlichkeit auch weiterhin häufig in den folgenden Texten vorzukommen. Der Lernprozeß folgt der gegebenen Reihenfolge und findet somit schneller zu mergende Zustände.

11.5.2.4.3.4.3 Festlegen der Tokenizer-Parameter...

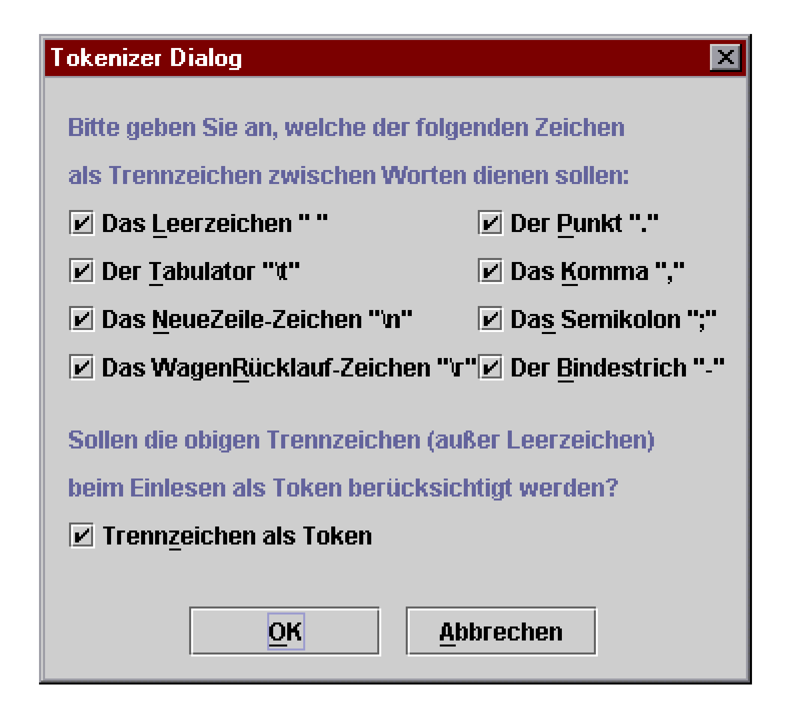


Abbildung 65 Festlegen der Tokenizer-Parameter

Mittels dieses Dialoges können die Zeichen angegeben werden, welche als Trennzeichen zwischen einzelnen Worten fungieren sollen, damit diese als Tokens verarbeitet werden können.

Zusätzlich zu der Angabe, welche Zeichen als Trennzeichen dienen, kann in diesem Dialog auch festgelegt werden, ob die Trennzeichen selbst als Token betrachtet werden sollen.

Ist das Auswahlfeld *Trennzeichen als Token* nicht selektiert, so werden die ausgewählten Zeichen zwar als Trennzeichen der Tokens betrachtet, nicht jedoch in den DFA aufgenommen. Eine etwaige Formatierung durch Steuerzeichen geht hierbei verloren.

11.5.2.4.3.4.4 Ersetzen von Zahlen

Das Ersetzen von Zahlen ist eine zusätzliche Option von *mAGENTa*. Hierbei werden im Vorfeld des eigentlichen Lernvorganges sämtliche Zahlen durch das Wort „XXX“ ersetzt. Dies hat zur Folge, daß der entstehende DFA und damit das entsprechende TokenComponent-Interface wesentlich kompakter wird. Das in [Goan et al. 1996] beschriebene Problem des „bushy prefix-trees“, also die Erzeugung von Zuständen mit hohem Fan-Out, kann dadurch verringert werden. Dies führt zu einem entsprechend übersichtlicheren TokenComponent-Interface.

Der beschriebene Effekt soll anhand der nachfolgenden Befunddaten verdeutlicht werden:

```
# Untersuchung ## 02.02.1998:Elektronenstrahltomographie des Herzens.
# Untersuchung ## 19.07.1999:Elektronenstrahltomographie des Herzens.
# Untersuchung ## 12.12.1999:Elektronenstrahltomographie des Herzens.
# Untersuchung ## 03.04.1998:Elektronenstrahltomographie des Herzens.
```

Der Mergingvorgang erfolgt dabei auf Basis der Regelmenge von Angluin (Regeln 1, 2a und 2b) und einem k-Vorgängerwert von 2.

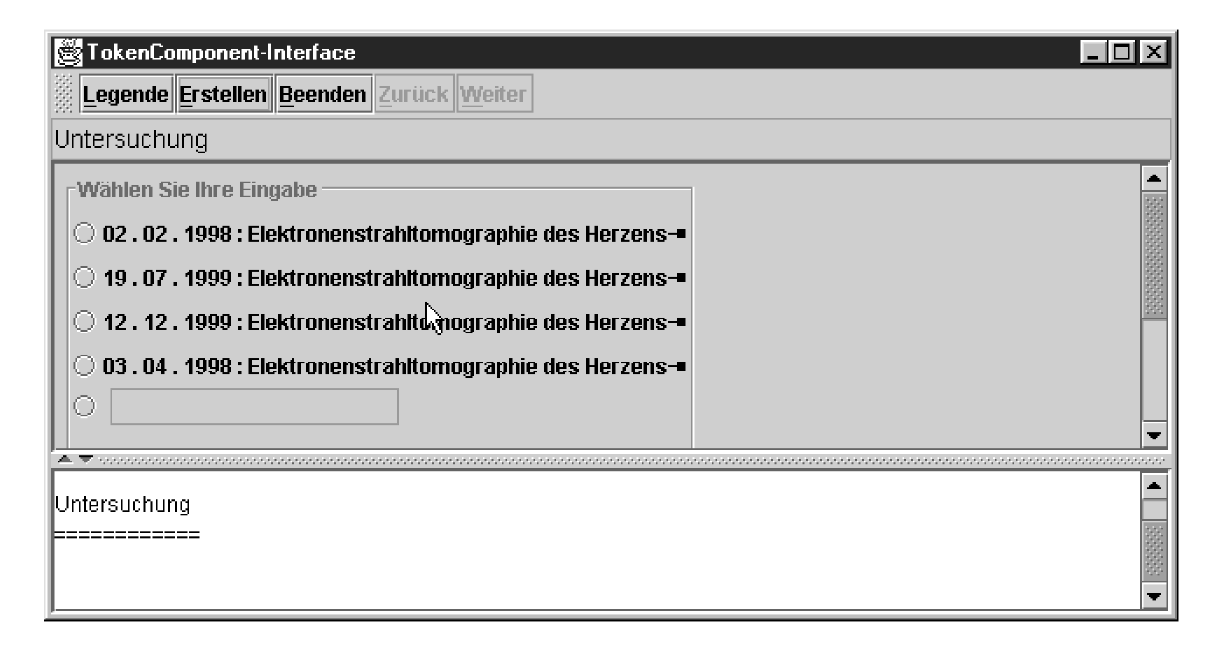


Abbildung 66 TCI nach Lernvorgang ohne vorherige Ersetzung von Zahlen

In Abbildung 66 wird das oben beschriebene Phänomen bereits nach wenigen Befunddaten deutlich.

Im Gegensatz dazu führt eine Ersetzung der Zahlen im Vorfeld des Mergingvorganges durch die Generalisierung des Automaten zu einer übersichtlicheren und benutzerfreundlicheren Oberfläche (siehe Abbildung 67).

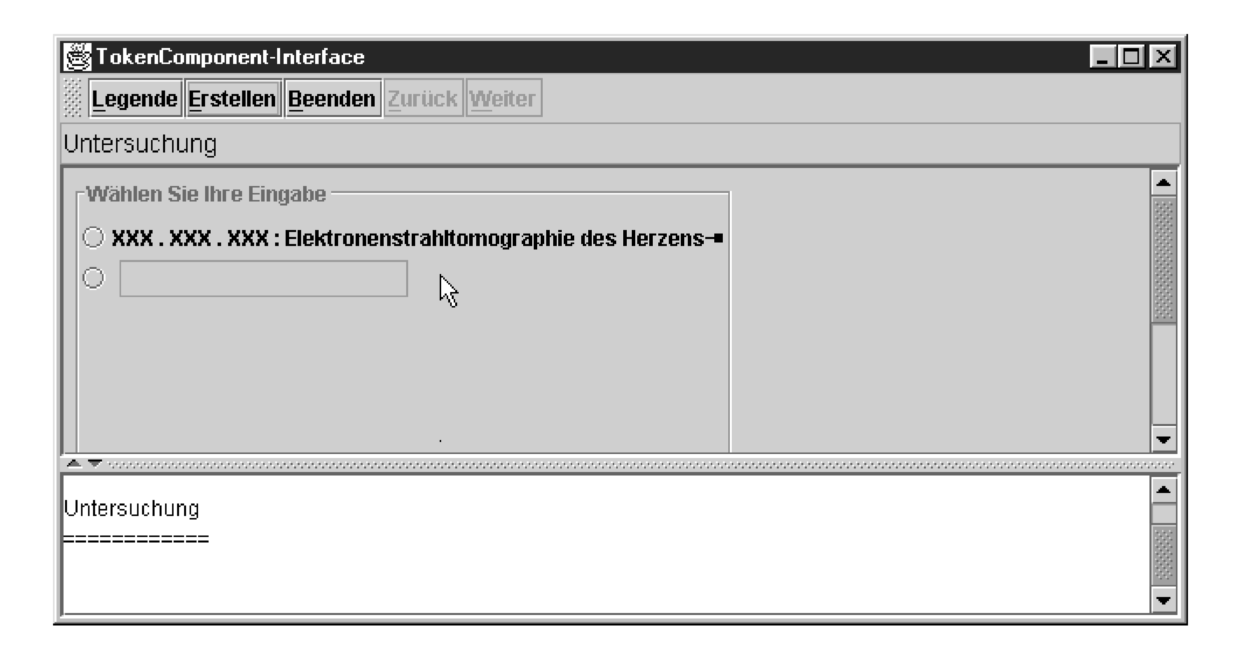


Abbildung 67 TCI nach einem Lernvorgang mit vorheriger Ersetzung von Zahlen

Am Ende des Befundungsvorganges hat der Benutzer dann die Möglichkeit, die korrekten Zahlen für den Platzhalter in den erstellten Text einzusetzen (siehe Kapitel 11.5.7 "Das Befund-Fenster").

11.5.2.4.3.4.5 Festlegen der Delimiter...

Damit der Lernalgorithmus erkennen kann, wann ein MetaAbschnitt beginnt und wie dieser Abschnitt betitelt ist, muß vor dem eigentlichen Abschnitt der MetaTitel stehen. Dieser MetaTitel besteht aus dem Start-Delimiter, dem eigentlichen MetaTitel und dem End-Delimiter. Er dient somit in der Vorverarbeitung des Lernvorganges dazu statt eines „normalen“ Zustandes einen MetaZustand zu generieren (siehe Kapitel 8.3.1).

So führt z. B. der folgende Eintrag in einem Befundtext dazu, daß ein MetaZustand mit dem Titel "Untersuchung" erzeugt wird, wobei die nachfolgenden Worte als entsprechender Abschnittstext interpretiert werden:

```
# Untersuchung ## 02.02.1998:Elektronenstrahltomographie des Herzens.
```

In diesem Beispiel ist also „#“ der Start-Delimiter, „##“ der End-Delimiter, „Untersuchung“ der MetaTitel des neuen MetaAbschnittes und „Elektronenstrahltomographie des Herzens“ der zugehörige Abschnittstext.

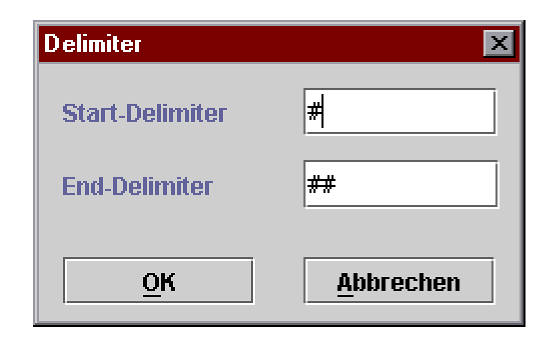


Abbildung 68 Festlegen der Delimeter

Wie aus der obigen Abbildung ersichtlich, kann in den sich bei Anwahl des Menüpunktes öffnenden Dialog jeweils in die Textfelder der passende Delimiter eingetragen werden. Anschließend muß die Eingabe mit Drücken des *OK*-Knopfes bestätigt werden.

11.5.2.4.3.4.6 Look & Feel...

Eins der herausragenden Merkmale von der Java 2 Platform (insbesondere Swing) ist die Möglichkeit des „Pluggable Look & Feel“, der zur Laufzeit auswechselbaren Darstellung von Java-Programmen.



Abbildung 69 Look & Feel

Nach Betätigung dieses Menüpunktes erscheint der obige Dialog, mit dem der Benutzer zwischen den angegebenen „Look & Feels“ wählen kann. Wird der Auswahlknopf *Windows* betätigt, so erscheint *mAGENTa* gemäß dem Style-Guide von Windows 95, wird *CDE/Motif* betätigt wechselt *mAGENTa* wieder entsprechend. Dabei ist es unerheblich, unter welchem Betriebssystem *mAGENTa* gerade läuft. Voreingestellt ist das Java „Look & Feel“ *Metal*, das auch bei den abgebildeten Bildschirmdarstellungen verwendet wurde.

11.5.2.4.3.4.7 Internet-Verbindung vorhanden

Durch dieses Auswahlfeld wird festgelegt, wie *mAGENTa* sich bezüglich der Dateiablage verhalten soll. Ist das Feld selektiert, so versucht *mAGENTa* zum Öffnen von Dateien auf das Netzwerk zuzugreifen. Ein Speichern von Dateien ist aufgrund von Sicherheitsbeschränkungen nicht erlaubt. Anzumerken wäre, daß es natürlich auch möglich ist lesend auf das lokale Dateisystem zuzugreifen. (siehe Kapitel 11.5.2.3.2.2 "Öffne externe Datei aus dem Netzwerk").

Ist das Auswahlfeld nicht selektiert, so wird bei dem Öffnen und Speichern von Dateien auf das lokale Dateisystem zugegriffen (siehe Kapitel 11.5.2.3.2.1 "Öffne lokale Datei" bzw. 11.5.2.4.1.4 "Speichern").

11.5.2.4.4 Das Hilfe-Menü

Das *Hilfe*-Menü hat zwei Einträge. Die Punkte *Hilfe* und *Info*.

11.5.2.4.4.1 Hilfe

Sobald dieser Menüpunkt angewählt wird, erscheint in dem im Hintergrund des Hauptfensters laufenden Browser der Hilfetext zu *mAGENTa*.

Dies ist ein ähnliches Vorgehen wie die Darstellung der Indexdateien bei dem Öffnen von Dateien über das Netzwerk (siehe Kapitel 11.5.2.3.2.2 "Öffne externe Datei aus dem Netzwerk").

Durch den Hilfetext werden die für die Benutzung wichtigen Funktionen erklärt und dem Benutzer ein ausführliches Nachschlagewerk über *mAGENTa* zur Hand gegeben. Durch diese Online-Hilfe ist der Benutzer nicht mehr darauf angewiesen, während der Benutzung des Systems ein gedrucktes Handbuch zur Verfügung zu haben.

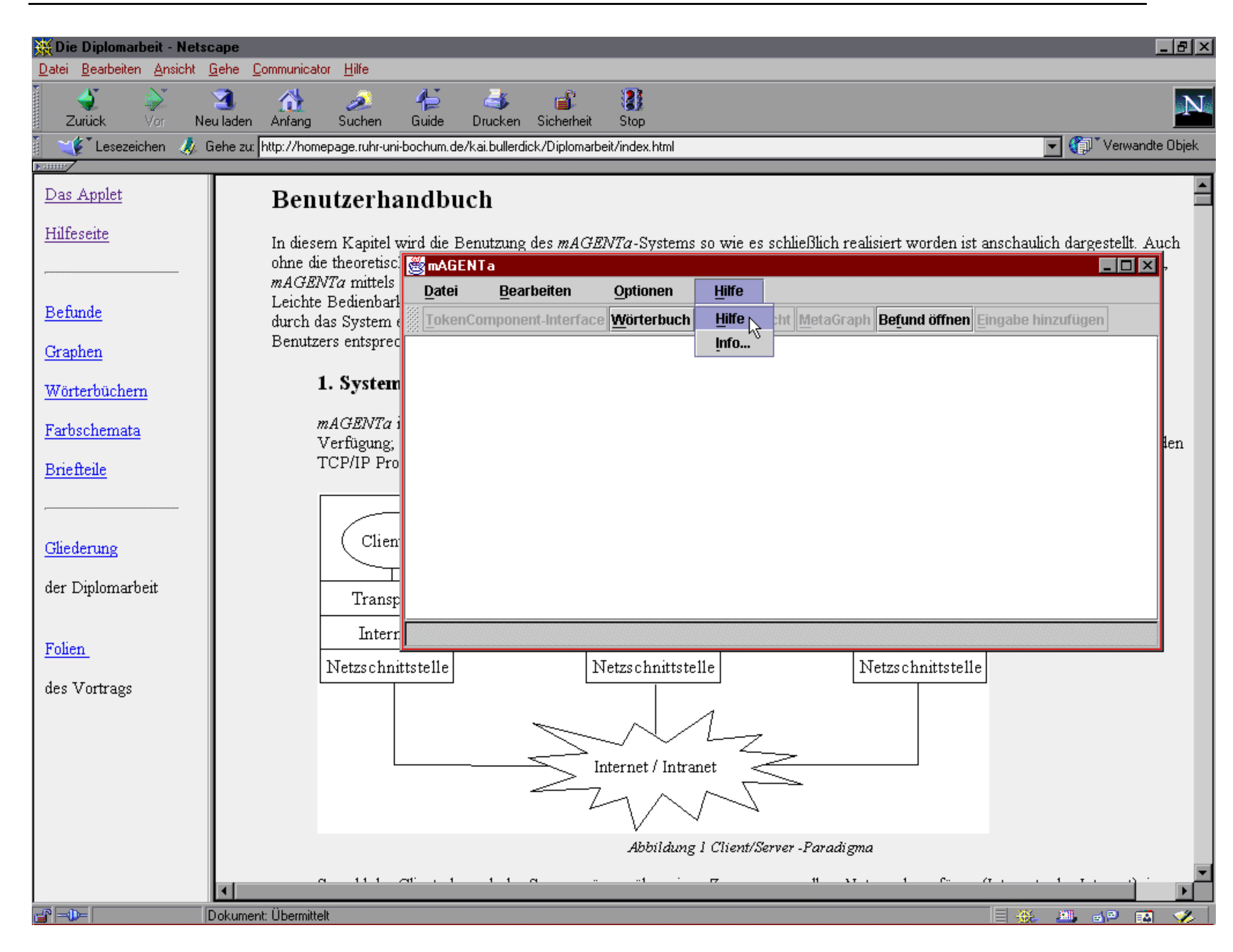


Abbildung 70 Der Hilfetext

Als Hilfetext wird das zu HTML konvertierte und überarbeitete Benutzerhandbuch verwendet.

11.5.2.4.4.2 Info...

Durch Betätigung dieses Menüpunktes öffnet sich ein Dialog, der kurz den Titel und die Autoren von *mAGENTa* vorstellt. Durch Betätigung des *OK*-Knopfes wird der Dialog wieder geschlossen.

11.5.3 Der Graph

Der Graph ist die visuelle, animierte Darstellung des DFA. Es werden die Zustände, die von ihnen ausgehenden Kanten und die Tokens dargestellt. Für eine bessere Ansicht werden die Kanten, auf denen die Tokens liegen, aufgeteilt in Kanten, die von einem Zustand zu einem Token führen, und Kanten, die von einem Token zu einem Zustand führen.

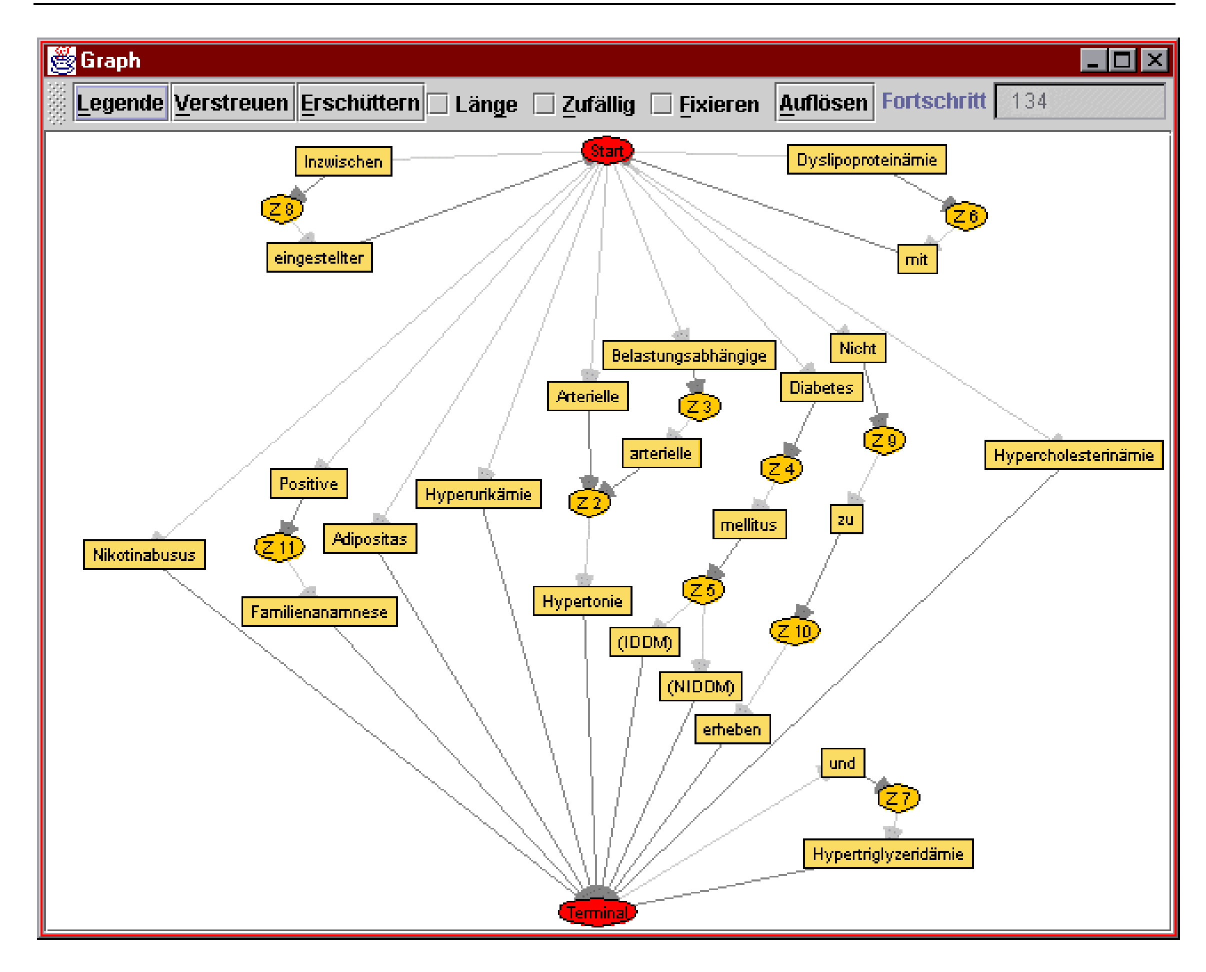
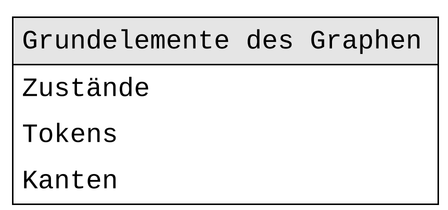


Abbildung 71 Graph-Fenster

Die obige Abbildung zeigt ein typisches Graph-Fenster. Am oberen Rand ist die Werkzeugleiste mit ihren verschiedenen Knöpfen zu sehen, darunter die Darstellung des DFA.

11.5.3.1 Die Elemente des Graphen

Der Graph ist aus drei Grundelementen aufgebaut, die jeweils unterschiedlich dargestellt werden.



Grundelemente des Graphen
Zustände
Tokens
Kanten

Tabelle 22 Grundelemente des Graphen

Wie in Abbildung 71 zu sehen, werden die verschiedenen Elemente des Graphen jeweils farbig dargestellt. Diese Farbgebung ist nach individuellen Gesichtspunkten einstellbar (siehe Kapitel 11.5.2.4.3.1.3 "Farbauswahl..."). Eine Erklärung der unterschiedlichen Farben der Abbildung ist in dem Kapitel 11.5.3.3.1 "Legende" zu finden.

Zustände und Tokens werden im Graph auch als Knoten bezeichnet.

Der Graph bietet nicht nur eine umfassende Gesamtdarstellung des DFA, sondern kann auch gezielt Auskunft über einzelne Zustände und Tokens geben. Wird ein Knoten mit der sekundären (i.A. rechten) Maustaste angeklickt, so öffnet sich ein spezifisches Kontextmenü, das weitere Informationen über diesen Knoten bietet und auch einige nur für diesen Knoten gültige Funktionen zur Verfügung stellt.

Auch können die Knoten des Graphen einzeln frei auf dem Bildschirm mittels der „Drag & Drop“-Technik von einer Position auf die nächste gezogen werden. Dies bedeutet, daß ein Knoten, der mit der primären (i.A. linken) Maustaste angeklickt wird, dann bei gedrückter Maustaste an eine beliebige Position innerhalb des Fensters gezogen werden kann. Da der Graph während dieser Aktion weiterhin versucht, sich gleichmäßig in dem Fenster zu verteilen, werden auch diejenigen Knoten, die mit dem selektierten durch Kanten verbunden sind, versuchen, den „optimalen“ Abstand zu erhalten und folgen somit dem selektierten Knoten. Sie werden „mitgezogen“.

Zur Auswahl eines Knotens ist es nicht notwendig, diesen Knoten exakt mit dem Mauszeiger zu treffen, sondern es reicht, in die Nähe des Knoten zu klicken. *mAGENTa* selektiert dann automatisch den nächstgelegenen Knoten.

11.5.3.1.1 Zustände

Zustände des DFA werden als Ovale dargestellt. Beschriftet sind sie jeweils mit „Z“ und einer fortlaufenden Nummer. Optisch besonders hervorgehoben sind der StartZustand, sowie die Meta- und TerminalZustände. Der StartZustand befindet sich immer in der Mitte des oberen Fensterrandes, die TerminalZustände frei am unteren. MetaZustände verteilen sich ebenso wie die anderen Zustände gleichmäßig innerhalb des Fensters.

11.5.3.1.1.1 Kontextmenü eines Zustands

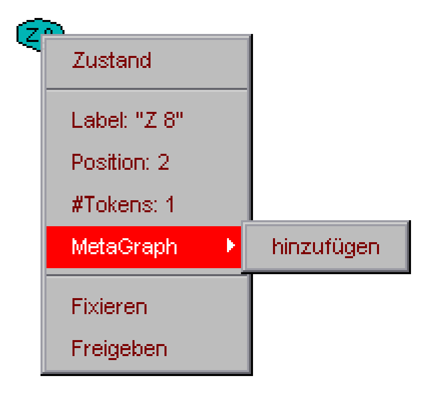


Abbildung 72 Kontextmenü eines Zustands

Das Kontextmenü eines Zustandes bietet folgende Einträge:

Menüpunkt	Beschreibung
Zustand	Der Typ des Knotens
Label	Die Beschriftung des Knotens, wie sie im Graph angezeigt wird
Position	Die Position innerhalb des DFA (interne Numerierung der Zustände)
#Tokens	Die Anzahl der nachfolgenden Tokens, d.h. die Anzahl der Zustandsübergänge von diesem Zustand aus
MetaGraph hinzufügen	Der Punkt des Untermenüs *MetaGraph* dient zum Hinzufügen des Zustandes zu dem MetaGraphen. Es wird ein Dialog geöffnet, in den der MetaTitel eingetragen werden kann
Fixieren	Fixieren des Knotens an seiner aktuellen Fensterposition. Bei der Animation des Graphen bewegt sich dieser Knoten nicht mehr mit
Freigeben	Freigeben eines fixierten Knotens. Ein ehemals fixierter Knoten bewegt sich wieder gemäß des Animationsalgorithmus des Graphen

Tabelle 23 Kontextmenü eines Zustandes

11.5.3.1.1.2 Kontextmenü eines MetaZustands

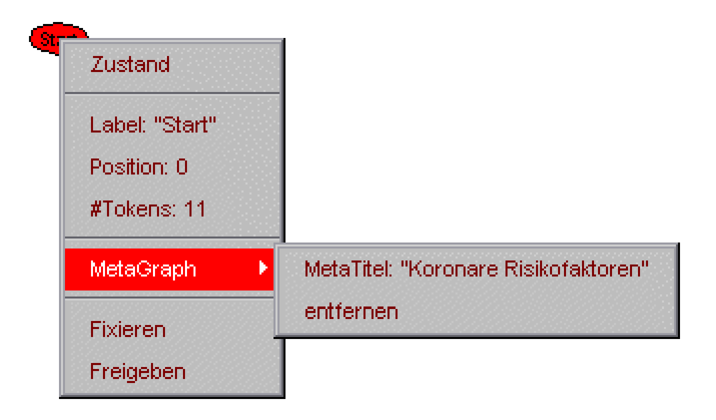


Abbildung 73 Kontextmenü eines MetaZustandes

Das Kontextmenü eines MetaZustandes ist wie das eines normalen Zustandes aufgebaut, mit die Unterschied, daß das Untermenü *MetaGraph* zwei andere Punkte enthält:

Menüpunkt	Beschreibung
MetaGraph MetaTitel	Der MetaTitel des Zustandes, so wie er in der MetaGraph-Ansicht erscheint
MetaGraph entfernen	Entfernen der MetaZustands-Eigenschaft. Ein MetaZustand wird zu einem normalen Zustand und aus dem MetaGraph entfernt

Tabelle 24 Kontextmenü eines MetaZustandes

11.5.3.1.2 Tokens

Tokens, die einen Zustand in den nächsten überführen, sind als Rechtecke zu erkennen. Beschriftet sind sie jeweils mit dem Wort, das den vorhergehenden Zustand in den folgenden überführt.

11.5.3.1.2.1 Kontextmenü eines Tokens



Abbildung 74 Kontextmenü eines Tokens

Das Kontextmenü eines Tokens bietet folgende Einträge:

Menüpunkt	Beschreibung
Token	Der Typ des Knotens
Aufschrift	Die Beschriftung des Knotens, wie sie im Graph angezeigt wird
Position	Die Position innerhalb des DFA (interne Numerierung der Tokens)
Frequenz	Die Frequenz des Tokens, d.h. die Anzahl wie häufig dieses Token gelernt bzw. im TCI angeklickt wurde
Fixieren	Fixieren des Knotens an seiner aktuellen Fensterposition
Freigeben	Freigeben eines fixierten Knotens

Tabelle 25 Kontextmenü eines Tokens

11.5.3.1.3 Kanten

Kanten sind die Verbindungsstellen zwischen den einzelnen Knoten. Es wird unterschieden zwischen Kanten, die von einem Zustand zu einem Token führen und Kanten, die von einem Token zu einem Zustand führen. Diese zusätzliche Unterscheidung erleichtert es, einen Pfad im Graph zu verfolgen. Deshalb verfügen die Kanten zusätzlich über eine Pfeilspitze, die auf das Zielelement der Kante zeigt.

11.5.3.2 Die Animation des Graphen

Da der Graph mitunter sehr komplex werden kann, ist es notwendig, daß dieser in einer geordneten Darstellung angezeigt wird. Der Graph versucht, in diese geordnete Darstellung zu gelangen, indem sich die einzelnen Knoten (Zustände und Tokens) gleichmäßig in dem Fenster verteilen und danach streben, jeweils einen bestimmten Abstand zueinander zu erlangen.

Dieser „optimale“ Abstand kann mittels der Option *Angestrebte Kantenlänge* festgelegt werden (siehe Kapitel 11.5.2.4.3.1.2 "Angestrebte Kantenlänge...").

Der Graph ist in einer geordneten Darstellung, wenn kein Knoten von einem anderen überdeckt wird, und die Kanten keine Überschneidungen aufweisen. Dieses Ziel ist in realen Graphen meistens nicht erreichbar, da einerseits nicht jeder beliebige Graph planar ist [Wegener, 1996] und andererseits die Fenstergröße einfach zu klein ist, um alle Knoten anzuzeigen. Der Graph strebt solange nach einer geordneten Darstellung, bis der Benutzer den Prozeß anhält, da eine befriedigende Darstellung erreicht worden ist (insbesondere dann, wenn nur Teile des gesamten Graphen von primärem Interesse sind).

Die folgende Skizze des Algorithmus stellt das Vorgehen informal dar:

Der Graph-Algorithmus
Solange kein Anhalten durch den Benutzer: Für jeden Knoten wird der Abstand zu seinen direkten Vorgängern und Nachfolgern bestimmt.   Schrittweise versuchen diese Nachbarn, zwischen sich den „optimalen“ Abstand zu erreichen.   Ist der „optimale“ Abstand größer als der tatsächlich, bewegen sich die Knoten schrittweise aufeinander zu.   Ist der „optimale“ Abstand kleiner, so entfernen sie sich voneinander.

Tabelle 26 Der Graph-Algorithmus

Aus Gründen der Effizienz wird bei der Darstellung des Graphen die Technik des „Double-Buffering“ verwendet (u. a. [Krüger, 1999]). Für jeden Knoten wird erst dessen neue Position berechnet, dieser Knoten dann an seiner neuen Position in ein „Offscreen-Image“ geschrieben, und erst wenn alle Knoten einmal berechnet und geschrieben worden sind wird dieses „Offscreen-Image“ auf einmal in die Fensteroberfläche kopiert. Im Sinne des obigen Algorithmus wird dies als ein Animationsschritt betrachtet.

Die Schrittweite, mit der sich die Knoten jeweils bewegen, beträgt 5 Pixel. Dieser Wert ermöglicht eine weiche („smooth“) Animation der Knotenbewegung, bei gleichzeitig hoher Animationsgeschwindigkeit.

Der Graph versucht, die gesamte Fläche des Fensters auszunutzen, und paßt sich automatisch der Größe des Fensters an, wenn diese sich ändern sollte.

11.5.3.3 Die Werkzeugleiste

Die Werkzeugleiste bietet hauptsächlich Funktionen, die eine bessere Ansicht des Graphen bieten. Diese Funktionen haben jeweils Auswirkungen auf den gesamten Graphen und nicht nur auf einzelne Knoten so wie die Kontextmenüs. Außerdem können von hier aus weitere Informationen über den Graph eingesehen werden.

11.5.3.3.1 Legende

Die Betätigung dieses Knopfes öffnet einen Dialog, der die Darstellung der einzelnen Elemente des Graphen erläutert.

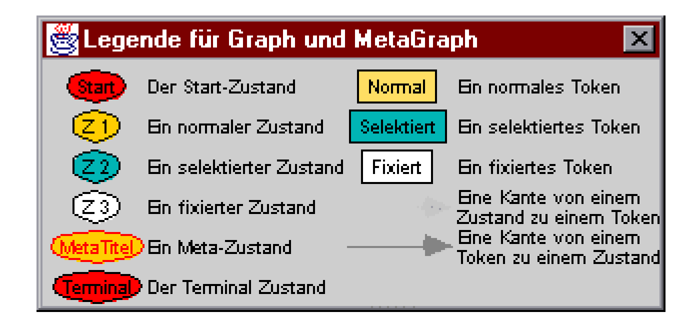


Abbildung 75 Legende des Graphen

Auf der linken Seite des Dialoges sind die verschiedenen Darstellungsformen der Zustände zu sehen, auf der rechten die der Tokens und darunter die der Kanten. Die unterschiedliche Farbgebung dient zur besseren Unterscheidung der Elemente und kann jeweils individuell eingestellt werden (siehe Kapitel 11.5.2.4.3.1.3 "Farbauswahl..."). Wird die Farbe eines Elementes geändert, so ändert sich synchron dessen Darstellung in dem Graphen und auch in der Legende.

11.5.3.3.2 Verstreuen

Der Knopf *Verstreuen* dient dazu, die Knoten des Graphen zufällig innerhalb des Fensters zu verteilen.

Eine zufällige Verteilung der Knoten bietet sich an, um nach Aufruf des Graph-Fensters eine ungeordnete, aber gleichmäßige Verteilung zu erlangen. Anschließend kann der Animations-Algorithmus besser die Ordnung des Graphen herbeiführen, als wenn die Knoten zu Beginn nur in einem kleinen Bereich des Fensters wären.

11.5.3.3.3 Erschüttern

Eine Betätigung dieses Knopfes führt zu einer kurzen „Erschütterung“ des Graphen. Sämtliche Knoten werden jeweils zufällig um bis zu 40 Pixel in der x- und y-Achse bewegt.

Die Verwendung dieser Funktion ist sinnvoll, wenn der Animations-Algorithmus bei der Verteilung der Knoten nicht voran schreitet, da die Knoten sich in einer ungünstigen Lage zueinander befinden. Z. B. zwei benachbarte Knoten müssen sich aneinander vorbei bewegen (der eine von links nach rechts, der andere rechts nach links) können dies aber nicht, da sie sich gegenseitig abstoßen. Die Betätigung des Knopfes führt dann dazu, daß die Position der Knoten sich zufällig verändert und sie sich somit besser aneinander vorbei bewegen können.

11.5.3.3.4 Länge

Ist dieses Auswahlfeld selektiert, so wird an die Kanten des Graphen jeweils deren Länge geschrieben. Die Länge ist in Pixeln angegeben.

Diese Angabe hilft dem Benutzer, eine für die Größe des Graphen und des Fensters passende Länge der Kanten (siehe Kapitel 11.5.2.4.3.1.2 "Angestrebte Kantenlänge...") einzugeben, damit sich die Knoten gleichmäßiger in dem Graph-Fenster verteilen.

11.5.3.3.5 Zufällig

Solange dieses Auswahlfeld selektiert ist, wird pro Animationsschritt ein zufällig ausgewählter Knoten um eine zufällige Anzahl von Pixeln (bis zu 40 von der ursprünglichen Position entfernt) in der x- und y-Achse bewegt.

Dies hat die gleiche Zweckmäßigkeit wie die *Erschüttern*-Funktion, ist aber von beliebiger Dauer.

11.5.3.3.6 Fixieren

Durch Selektion dieses Auswahlfeldes signalisiert der Benutzer, daß die Animation des Graphen angehalten werden soll. Jeder einzelne Knoten des Graphen wird an seiner aktuellen Position festgesetzt, seine Farbe verändert sich zu der in der Farbauswahl festgelegten Farbe für die fixierten Knoten.

Dies ist nützlich, um einen schon in ausreichendem Maße im Fenster verteilten Graphen daran zu hindern sich weiter zu bewegen. Der Benutzer kann nun, da die Knoten fixiert sind, gezielter einzelne selektieren, per „Drag & Drop“ an eine gewünschte Position ziehen oder sich die Kontextmenüs anzeigen lassen.

11.5.3.3.7 Auflösen

Bei einer großen Anzahl von Knoten des Graphen kann es eine längere Zeit dauern, bis eine geordnete Darstellung der Knoten vorliegt. Dies ist hauptsächlich von der Geschwindigkeit, mit der die Elemente gezeichnet werden, abhängig und nicht von der Berechnung der neuen Positionen der Knoten.

Deshalb ist es möglich *mAGENTa* anzuweisen, den Animations-Algorithmus zur Berechnung der neuen Positionen ohne gleichzeitige Darstellung des Graphen auszuführen.

Durch Betätigung des *Auflösen*-Knopfes wird eine proportional zur Größe des Graphen liegende Anzahl von Animationsschritten ausgeführt, ohne die Elemente jeweils an ihren neuen Positionen zu zeichnen. Zusätzlich wird die übliche Schrittweite der Knoten auf 20 Pixel vervierfacht, damit die Knoten schneller ihren Weg zu einer geordneten Darstellung begehen.

Die Anzahl der Animationsschritte, die im Hintergrund ausgeführt werden, ist gleich der Anzahl der Knoten, also gleich der Summe der Zustände und Tokens. Eine proportionale Anzahl von Schritten bringt den Graphen näher an eine geordnete Darstellung als eine feste Anzahl, da ein Graph mit einer großen Anzahl von Knoten mehr Animationsschritte für die Ordnung benötigt als einer mit einer geringeren. Außerdem könnten bei einer geringen Anzahl von Knoten zu viele Schritte getätigt werden, obwohl der Graph sich schon in einer geordneten Darstellung befindet.

Sind entsprechend viele Schritte berechnet worden, so werden schließlich die Knoten und Kanten an ihren aktuellen Positionen gezeichnet und der Animations-Algorithmus fährt wieder fort.

11.5.3.3.8 Fortschritt

Zur Information des Benutzers wird in diesem Textfeld die Anzahl der bisher berechneten Animationsschritte festgehalten. Bei jedem weiteren Schritt wird diese Anzahl inkrementiert.

Wird während der Animation der *Auflösen*-Knopf betätigt, so erscheint hier eine Prozentzahl, die anzeigt wie weit die Berechnungen ohne Aktualisierung der Graphansicht fortgeschritten sind. Sind diese Berechnungen durchgeführt, so wird wieder die gesamte Anzahl der Animationsschritte als absolute Zahl dargestellt.

11.5.4 Der MetaGraph

So wie der Graph die Zustände und Tokens des DFA anzeigt, so zeigt der MetaGraph die MetaZustände an. Die Darstellung und Funktionalität des MetaGraph-Fensters ist mit der des Graph-Fensters identisch, so daß hier nur auf die Unterschiede eingegangen wird.

11.5.4.1 Die Elemente des MetaGraphen

Im MetaGraph werden nur die MetaZustände des DFA dargestellt. Wie aus Abbildung 76 MetaGraph-Fenster ersichtlich, fehlen die Tokens und die übrigen Zustände. Dementsprechend gibt es nur eine Art von Kanten, nämlich diejenigen, die von einem MetaZustand zu dem nächsten MetaZustand führen. Abstrakt betrachtet befinden sich auf diesen Kanten die einzelnen MetaAbschnitte. Die Kante, die z.B. von dem MetaZustand *Koronare Risikofaktoren* zu dem MetaZustand *Untersuchungstechnik* führt, stellt den kompletten Abschnitt über die koronaren Risikofaktoren eines Patienten dar. In diesem Abschnitt gibt es also einen Zustand, der in den MetaZustand *Untersuchungstechnik* überführt werden kann.

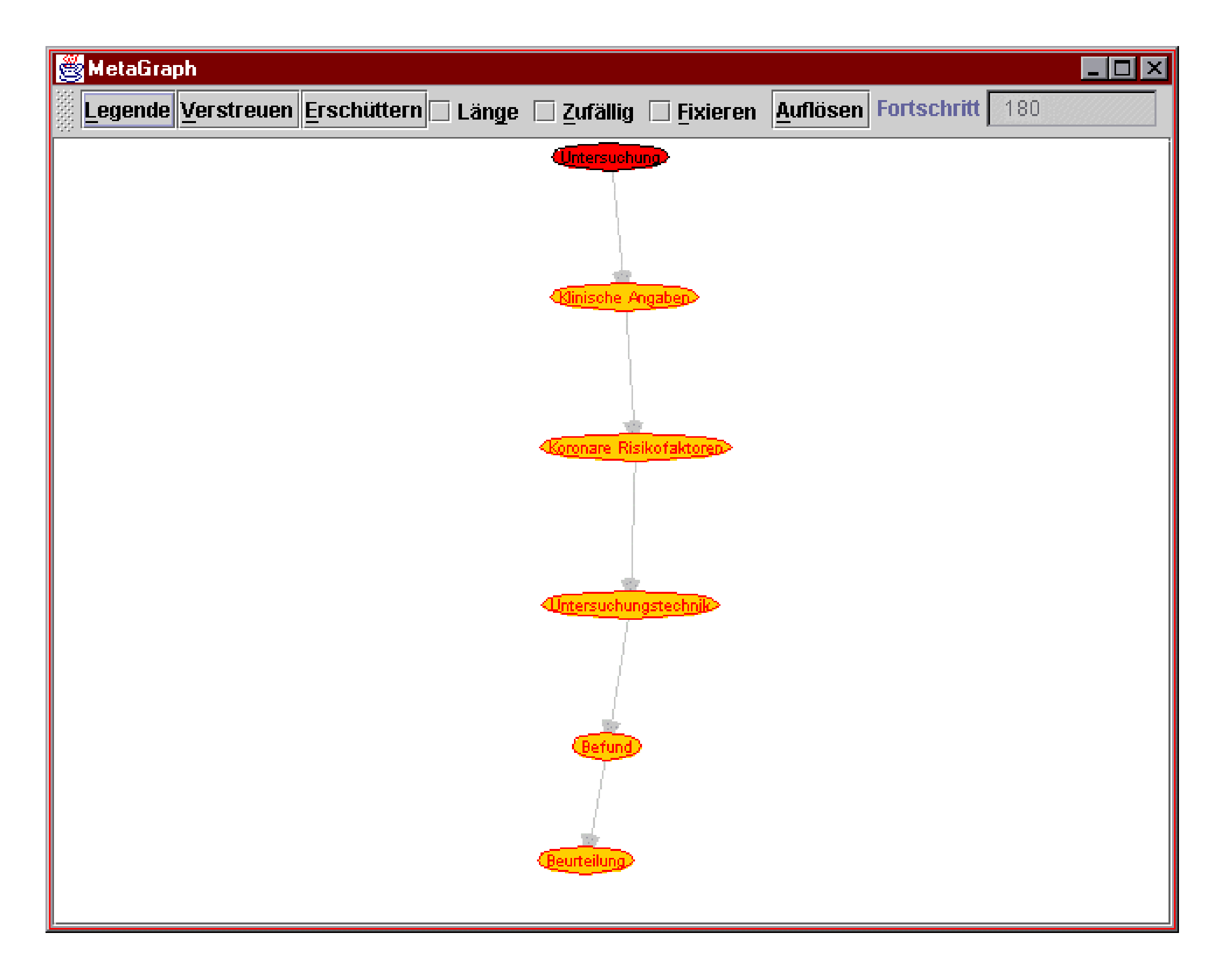


Abbildung 76 MetaGraph-Fenster

11.5.4.1.1 Kontextmenü eines MetaZustandes

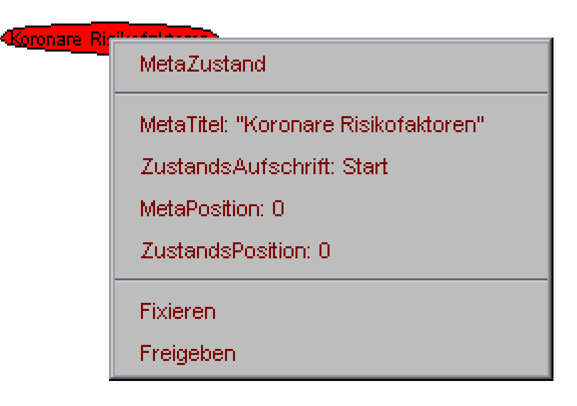


Abbildung 77 Kontextmenü eines MetaZustandes im MetaGraph

Das Kontextmenü eines MetaZustandes im MetaGraphen bietet folgende Einträge:

Menüpunkt	Beschreibung
Token	Der Typ des Knotens
MetaTitel	Der MetaTitel des Zustandes als Überschrift des MetaAbschnittes
ZustandsAufschrift	Die Beschriftung des Zustandes im Graphen
MetaPosition	Die MetaPosition innerhalb des DFA (interne Numerierung der MetaZustände)
ZustandsPosition	Die Position des Zustandes innerhalb des DFA (interne Numerierung der Zustände)
Fixieren	Fixieren des Knotens an seiner aktuellen Fensterposition
Freigeben	Freigeben eines fixierten Knotens

Tabelle 27 Kontextmenü eines MetaZustandes im MetaGraphen

11.5.5 Das Wörterbuch

Das Wörterbuch-Fenster zeigt die Einträge des Wörterbuches in sortierter Form an. Da mitunter viele verschiedene Worte in den gelernten Texten vorkommen, ist es nötig, diese in geeigneter Form darzustellen. Diese Form muß sowohl einen guten Überblick über sämtliche Einträge, als auch gezielten Zugriff auf einzelne bieten.

Da *mAGENTa* von verschiedenen Personen genutzt werden kann, ist es sinnvoll, wenn diese ihre einmal gelernten Texte austauschen und kombinieren, um so ein besseres Gesamtergebnis zu erreichen. Ebenso wie bei Graphen ist es auch bei Wörterbüchern möglich, diese anderen nutzbar zu machen, indem ein Benutzer sein erstelltes

Wörterbuch abspeichert, und andere es öffnen. Prinzipiell stellt sich hier eine gewisse Unabhängigkeit der Wörterbücher gegenüber den Graphen dar.

Das Wörterbuch wird selbständig ohne weitere Aktionen des Benutzers erweitert. Wird ein neuer Text gelernt, ein Graph geladen oder ein Eintrag in eine EmptyComponent des TCI vorgenommen, so werden die Wörter, die bisher noch nicht im Wörterbuch zu finden waren, automatisch eingetragen. Der Benutzer kann nun das Wörterbuch aufrufen, um gezielt einzelne Wörter durch Synonyme zu ersetzen. Jedes Vorkommen des Wortes als Token im DFA wird nun durch das oder die neuen Wörter ersetzt. Werden neue Texte eingelesen, so werden diese automatisch auf Einträge des Wörterbuches überprüft und diese gegebenenfalls vor dem Lernprozeß ersetzt.

Dies ist nicht nur in der Form einer 1:n – Ersetzung möglich, sondern auch in der Form m:n. Dies bedeutet, daß Wortgruppen oder ganze Sätze durch andere benutzerdefinierte ersetzt werden können.

Durch domänenspezifische Wörterbücher wird der Lernvorgang unterstützt und so das Lernergebnis verbessert.

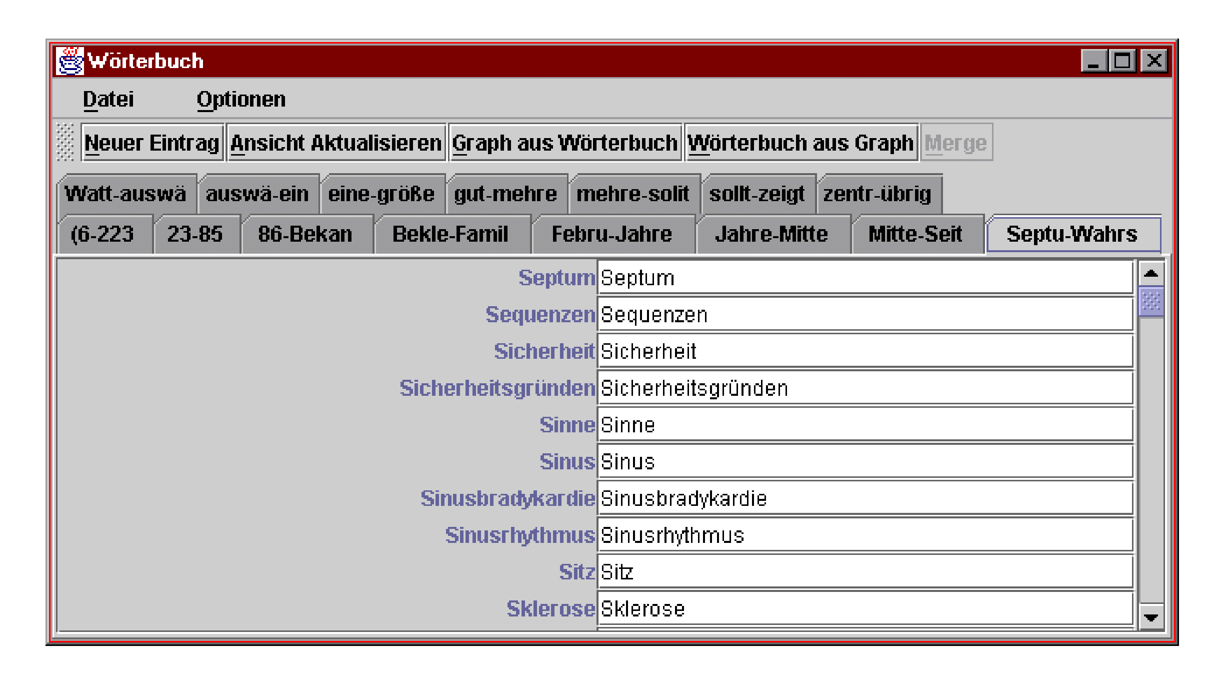


Abbildung 78 Wörterbuch-Fenster

Wie andere Fenster auch bietet das Wörterbuch-Fenster ein Datei- und Optionen-Menü sowie eine Werkzeugleiste für die wichtigsten Funktionen. Die Darstellung der Wörterbucheinträge selbst ist in der Mitte des Fensters zu sehen.

11.5.5.1 Die Wörterbucheinträge

Die Einträge des Wörterbuches sind gemäß des Unicode-Standards lexikographisch sortiert. Um einen Überfluß an Informationen zu vermeiden, kann das Wörterbuch in

einzelne Seiten eingeteilt werden. Der Benutzer kann vorgeben, wie viele Einträge pro Seite erwünscht sind (siehe 11.5.5.3.2.1 Maximale Anzahl von Einträgen pro Seite...), und *mAGENTa* teilt die Wörterbucheinträge in eine entsprechende Anzahl von Seiten ein.

Die einzelnen Seiten verfügen jeweils über einen Reiter (Tab), der anzeigt, welche Einträge sich auf ihr befinden. Es werden auf dem Reiter jeweils die ersten fünf Buchstaben des ersten und letzten Eintrages angezeigt. Durch Anwahl ihres Reiters kann die gewünschte Seite in den Vordergrund gebracht werden.

Die vertikal orientierten Einträge sind jeweils mit dem Wort und dessen Ersetzung aufgeführt. Links steht die Beschriftung, rechts ein Textfeld zur Aufnahme der Ersetzung. Zu Beginn ist das Textfeld mit dem Wort selbst gefüllt, da bisher der Benutzer noch keine Ersetzung eingetragen hat. Dies kann der Benutzer durch Überschreiben des Wortes vornehmen.

11.5.5.2 Die Werkzeugleiste

Die Werkzeugleiste bietet die Funktionalität, um das Wörterbuch zu erweitern und in Synchronisation mit dem Graphen bzw. DFA zu bringen.

11.5.5.2.1 Neuer Eintrag

Möchte der Benutzer Wörter ersetzen, die noch nicht in den bisherigen Texten vorkamen, wohl aber in zukünftigen, so können auch Einträge in das Wörterbuch vorgenommen werden, die erst später zur Anwendung kommen.

Wird der Knopf *Neuer Eintrag* betätigt, so werden zwei Textfelder vor dem ersten Eintrag der sichtbaren Seite eingefügt. In das linke trägt der Benutzer das oder die zu ersetzenden Wörter ein, in das rechte die gewünschte Alternative (evtl. auch aus mehreren Wörtern bestehend).

Dies kann auch dazu genutzt werden, einzelne Einträge des Wörterbuches, die sich im Graph in einer linearen Liste von Tokens befinden, durch ein gemeinsames Wort zu ersetzen.

11.5.5.2.2 Ansicht Aktualisieren

Dieser Knopf gleicht die Ansicht des Wörterbuches mit dessen Einträgen ab.

Wurde z.B. ein neuer Eintrag in das Wörterbuch vorgenommen, so wird das Wort und seine Ersetzung an der lexikographisch richtigen Stelle angezeigt.

11.5.5.2.3 Graph aus Wörterbuch

Wurden in dem Wörterbuch Ersetzungen vorgenommen oder neue Einträge hinzugefügt, so muß der Graph entsprechend aktualisiert werden, damit beide Fenster über synchrone Darstellungen verfügen.

Bei einer Ersetzung eines Wortes durch das Wörterbuch werden durch Betätigung dieses Knopfes sämtliche Tokens des Graphen, die mit diesem Wort beschriftet sind, durch Tokens, die mit der Ersetzung des Wortes beschriftet sind, ersetzt.

11.5.5.2.4 Wörterbuch aus Graph

Sind hingegen an dem DFA Veränderungen (i.A. durch Lernen von neuen Texten) vorgenommen worden, so kann es auch hier vorkommen, daß die Einträge des Wörterbuches nicht über die gleiche Aktualität wie die Einträge des DFA verfügen. Insbesondere, wenn das Wörterbuch-Fenster geöffnet ist, und dem DFA neue Texte hinzugefügt werden, werden in dem Wörterbuch-Fenster neu dazugekommene Worte nicht sofort angezeigt.

Wird dieser Knopf betätigt, so werden die Einträge des DFA in das Wörterbuch aufgenommen und die Ansicht anschließend aktualisiert.

11.5.5.2.5 Merge

Ist in dem Dialog für die Merging-Parameter (siehe 11.5.2.4.3.1.1 Festlegen der Merging-Parameter...) festgelegt worden, daß der Lernprozeß nicht sofort bei jeder Änderung am DFA gestartet werden soll, so ist dieser Knopf nach Änderung von Einträgen des Wörterbuches aktivierbar. Ansonsten wird bei jeder Änderung der Wörterbucheinträge direkt der Lernprozeß gestartet.

Möchte der Benutzer mehrere Änderungen an dem Wörterbuch vornehmen, so ist es oftmals sinnvoller, erst alle diese Änderungen einzutragen und dann gemeinsam auf einmal lernen zu lassen, indem durch Betätigung dieses Knopfes der Lernprozeß gestartet wird.

11.5.5.3 Das Menü

Das Menü verfügt im wesentlichen über die Funktionalität des Menüs des *mAGENTa*-Hauptfensters. Es gibt ein Menü für die Dateiverarbeitung und eins für Optionen. Da die Funktionalität sich sehr ähnelt, wird hier auf das entsprechende Kapitel verwiesen (siehe Kapitel 11.5.2.4 "Die Menüs").

11.5.5.3.1 Das Datei-Menü

11.5.5.3.1.1 Neu

Durch Anwahl dieses Menüpunktes werden die Einträge des Wörterbuches entfernt.

Ist ein DFA gelernt worden, so können durch Betätigung des Knopfes *Wörterbuch aus Graph* die Tokens des Graphen in das Wörterbuch eingetragen werden. Vorher eingetragene Ersetzungen sind allerdings nicht mehr in dem Wörterbuch vorhanden.

11.5.5.3.1.2 Öffnen...

Ein vorher abgespeichertes Wörterbuch kann hier wieder geöffnet werden. Je nach Einstellung erfolgt ein Dialog zum Öffnen einer lokalen Datei oder über das Netzwerk (siehe Kapitel 11.5.2.4.3.4.7 "Internet-Verbindung vorhanden").

Das vorher vorhandene Wörterbuch wird durch das neu geöffnete ersetzt.

11.5.5.3.1.3 Hinzufügen...

Im Gegensatz zum *Öffnen* eines Wörterbuches kann durch den Menüpunkt *Hinzufügen* ein Wörterbuch geöffnet werden, welches das bisher vorhandene nicht überschreibt, sondern mit ihm kombiniert wird. Die alten Einträge des Wörterbuches bleiben erhalten und werden durch zusätzliche aus dem gerade geöffneten ergänzt. So ist es möglich, daß verschiedene Benutzer ihre Wörterbücher kombinieren und somit eine gemeinsame Basis schaffen.

11.5.5.3.1.4 Speichern

Wurde das Wörterbuch schon einmal unter einem Namen gespeichert, so kann es mittels dieses Menüpunktes wieder direkt darunter lokal abgespeichert werden.

11.5.5.3.1.5 Speichern unter...

Dieser Menüpunkt öffnet einen Dateiauswahldialog, durch den das Wörterbuch unter einem einzugebenden Namen lokal abgespeichert werden kann.

11.5.5.3.1.6 Wörterbuch Liste anzeigen...

Um die Einträge des Wörterbuches in einer einfachen Form einzusehen, kann mittels dieses Menüpunktes ein Fenster geöffnet werden, daß ähnlich aufgebaut ist wie das Fenster für das Merging-Protokoll (siehe Kapitel 11.5.2.4.1.7 "Merging-Protokoll anzeigen...").

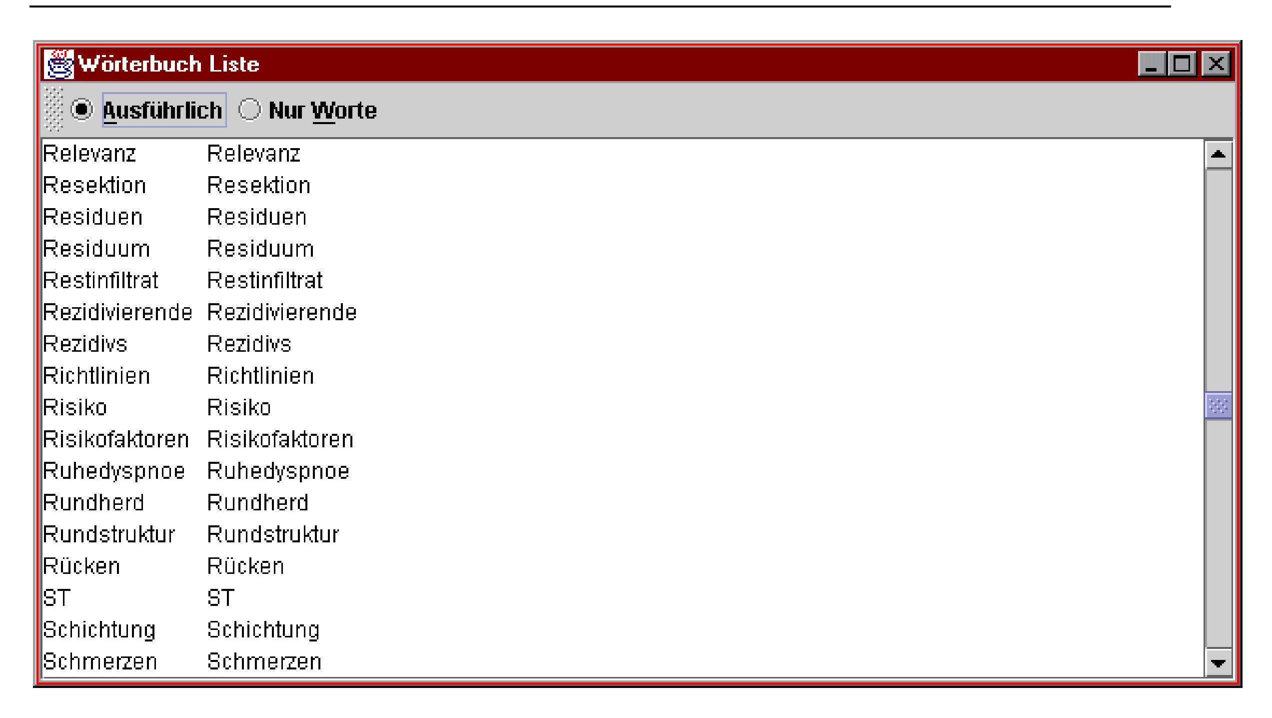


Abbildung 79 Wörterbuch Liste

Die Werkzeugleiste bietet zwei Auswahlknöpfe, mit denen zwischen den Ansichten umgeschaltet werden kann. Bei Auswahl des Knopfes *Ausführlich* werden sämtliche Einträge des Wörterbuches lexikographisch sortiert in dem Textfeld dargestellt, neben ihnen sind jeweils ihre Ersetzungen zu finden. Bei der Auswahl des Knopfes *Nur Worte* fehlen diese Ersetzungen, es werden nur die Worte selbst angezeigt.

Diese bloße Textdarstellung erlaubt es, die Einträge des Wörterbuches einfach zu exportieren, um sie weiterzuverarbeiten. Dies kann von Interesse sein, da es einen kompletten Überblick über alle Worte liefert, die in den bisher gelernten Texten vorkamen. Bei entsprechend vielen Texten entsteht somit eine vollständige Liste aller Worte einer spezifischen Domäne.

11.5.5.3.1.7 Beenden

Dieser Menüpunkt schließt das Wörterbuch-Fenster.

11.5.5.3.2 Das Optionen-Menü

11.5.5.3.2.1 Maximale Anzahl von Einträgen pro Seite...

Dieser Menüpunkt ist identisch mit dem Menüpunkt des Hauptfensters. Wenn er betätigt wird, öffnet sich derselbe Dialog.

Dieser Dialog bezieht sich auf die Anzahl der Einträge, die auf einer Seite des Wörterbuches angezeigt werden sollen.

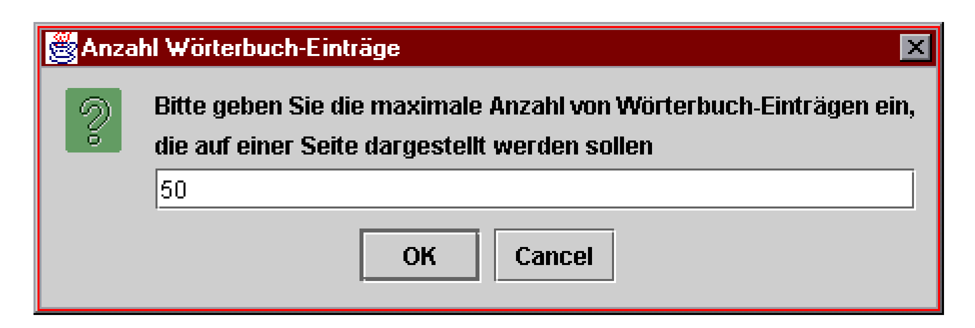


Abbildung 80 Maximale Anzahl von Einträgen pro Seite

Jede Seite kann eine festgelegte Anzahl von Wort-Einträgen und deren Ersetzung fassen. Diese Anzahl wirkt sich auf die Anzahl der Seiten des Wörterbuches aus, da gilt:

Anzahl der Worte / maximale Anzahl = Anzahl der Seiten

Die Anzahl der Einträge hat die Auswirkung, daß ein einzelner zu ändernder Eintrag durch den Benutzer schneller gefunden werden kann, da eine Wörterbuchseite jeweils mit den ersten fünf Buchstaben des ersten und letzten Eintrages betitelt ist.

Durch eine geringere Anzahl an Einträgen pro Seite wird außerdem die Geschwindigkeit erhöht, mit der die Wörterbuchseiten graphisch aufgebaut werden. Der voreingestellte Wert von 50 dürfte auf einem gewöhnlichen Personal-Computer keine merkbare Verzögerung bewirken.

Sollen alle Worte auf einer Seite angezeigt werden, so ist eine Zahl einzutragen, die größer oder gleich der Anzahl der Worte ist. Diese Anzahl läßt sich durch Aufruf des Graph-Eigenschaften Dialoges aus dem *mAGENTa* Hauptfenster (siehe Kapitel 11.5.2.4.1.8 "Eigenschaften des Graphen anzeigen...") ablesen.

11.5.6 Das TokenComponent-Interface

Das TokenComponent-Interface ermöglicht es aus einmal gelernten Texten weitere Texte jeweils auf eine effiziente Art zu erstellen. Es ist eine direkte Abbildung des DFA in eine benutzerfreundliche Darstellung.

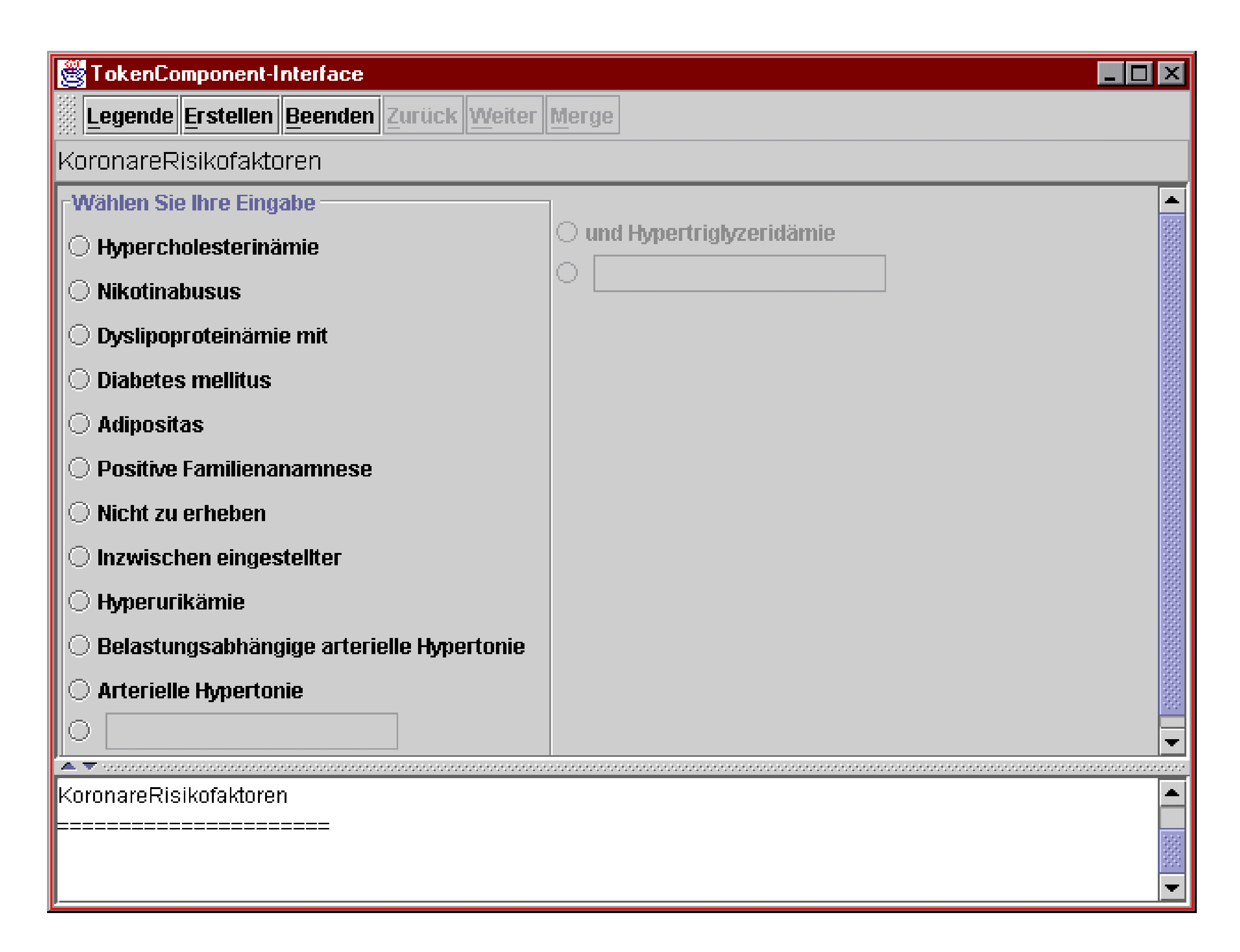


Abbildung 81 TokenComponent-Interface Fenster

Das Fenster verfügt am oberen Rand über eine Werkzeugleiste, darunter befindet sich ein Bereich für den Titel des aktuellen MetaAbschnittes. Den Hauptbereich des Fensters teilen sich der eigentliche Eingabebereich des TCI und darunter das Textfeld für den aktuell entstehenden Text. Diese beiden Bereiche sind durch einen horizontalen Mittelsteg getrennt. Dieser Mittelsteg ist beweglich, so daß die Größenverhältnisse der beiden Bereiche den Wünschen des Benutzers gemäß angepaßt werden können. Durch einfaches Ziehen mit der Maus kann der Mittelsteg verschoben werden. Wird einer der beiden kleinen Pfeile am linken Ende des Mittelsteges betätigt, so wird der Eingabebereich bzw. das Textfeld vergrößert, daß es die gesamte Fläche bedeckt. Durch erneute Anwahl wird wieder das ursprüngliche Größenverhältnis hergestellt.

Die Angaben über die MetaZustände gelten im folgenden nur, wenn die MetaStruktur auch aktiviert worden ist (siehe Kapitel 11.5.2.4.3.4.1 "MetaStruktur erwünscht").

11.5.6.1 Der MetaTitel

Wird bei der Arbeit mit *mAGENTa* die MetaStruktur verwendet, so erscheint in diesem Feld der Titel des aktuellen MetaAbschnittes zur Orientierung für den Benutzer.

11.5.6.2 Der Eingabebereich

In dem Eingabebereich werden die einzelnen TokenComponents dargestellt. TokenComponents ist der Oberbegriff für TokenButtons, TokenChoices und EmptyComponents. Die ersten beiden sind mit einer Aufschrift versehen; in die EmptyComponent kann Text, der dem DFA an dieser Stelle hinzugefügt werden soll, eingetragen werden.

11.5.6.2.1 Die TokenComponents

Das TCI zeigt die TokenComponents, die von einem Zustand bis zum nächsten MetaZustand aus erreichbar sind, an. Beginnend mit dem StartZustand des DFA werden für jede ausgehende Kante eines dargestellten Zustandes ein TokenButton (evtl. zusammengefaßt zu einer TokenChoice) und zusätzlich eine EmptyComponent angezeigt. In einer TokenComponent können auch mehrere Tokens konkateniert dargestellt werden, wenn sich die Tokens auf Kanten einer linearen Liste von Zuständen befinden.

11.5.6.2.2 Der ZustandsRahmen

Die TokenComponents, die der Benutzer aktuell auswählen kann (die direkt von dem aktuellen Zustand ausgehen), sind zur besseren Sichtbarkeit mit einem Rahmen versehen. Dessen Titel (*Wählen Sie Ihre Eingabe*) fordert den Benutzer auf, eine TokenComponent aus diesem Rahmen zu wählen.

TokenButtons werden wie gewöhnliche Auswahlknöpfe, TokenChoices wie Aufklapplisten bedient.

11.5.6.2.3 Eingabe eines neuen Textes in eine EmptyComponent

In EmptyComponents kann nach Selektion ihres Auswahlknopfes ein Text eingegeben werden. Dieser Text wird dann in den DFA als neuen Übergang dieses Zustandes eingefügt (siehe auch Kapitel 11.5.6.4.6 "Merge"). Der Eintrag in die EmptyCompontent führt durch die eingegebenen Tokens den aktuellen Zustand direkt in den nächsten MetaZustand.

Da oftmals schon vorhandene Texte nur geringfügig geändert werden sollen, ist es möglich eine TokenComponent als „Vorbild“ für einen neuen Eintrag zu wählen. Ist eine EmptyComponent selektiert, so kann anschließend eine TokenComponent angewählt werden. Deren Beschriftung wird in das Textfeld der EmptyComponent

eingetragen und der nachfolgende Zustand der TokenComponent dient nun auch als Nachfolger der EmptyComponent. Die beiden Tokens sind nun parallel in dem DFA angeordnet.

Ist die Eingabe in eine EmptyComponent vollständig, so muß zur Bestätigung der *Weiter*-Knopf betätigt werden.

11.5.6.2.4 Die TCI-Vorschau

Neben dem ersten Rahmen für den aktuellen Zustand können sich in horizontaler Anordnung noch weitere Rahmen befinden, die TokenComponents beinhalten. Diese sind allerdings noch nicht selektierbar, da die Eingabe gemäß der in dem DFA herrschenden Reihenfolge zu erfolgen hat, da sonst keine strukturierte Anleitung mehr erfolgen würde.

In Abbildung 81 findet sich neben dem aktuellen Rahmen ein zweiter, der einen TokenButton sowie eine EmptyComponent beinhaltet. Der Zustand, der durch den ersten Rahmen repräsentiert wird, wird gemäß des DFA über die Tokens der ersten TokenComponent in einen nachfolgenden Zustand überführt, der nun durch den zweiten Rahmen repräsentiert wird. Dies ist natürlich transitiv. Es werden so viele Rahmen angezeigt, bis ein Übergang zu einem Meta- oder TerminalZustand erreicht wird, oder für denselben Zustand noch ein Rahmen erstellt werden soll (Zyklus).

Gemäß den Voreinstellungen (siehe Anhang Voreinstellungen) sind die TokenComponents nach der Frequenz ihrer Tokens sortiert (siehe Kapitel 11.5.2.4.3.3.5 "Nach Frequenz"). Das wahrscheinlichste Token wird als erstes angezeigt. Also wird auch der nächste Rahmen für den Zustand angezeigt, der erreicht wird, wenn der Pfad über diese Tokens begangen wird. Möchte der Benutzer aber wissen, welche weiteren TokenComponents einer bestimmten TokenComponent folgen, die sich nicht an erster Stelle in dem Rahmen befindet, so ist lediglich die sekundäre Maustaste (i. A. die rechte) zu betätigen. Unmittelbar werden der nächste und evtl. folgende Rahmen durch den Rahmen ersetzt, der den Zustand repräsentiert, der erreicht wird, wenn die betrachtete TokenComponent betätigt wird. Auch hier werden evtl. weiter Rahmen erzeugt. Diese TCI-Vorschau ist nicht nur für den ersten Rahmen, sondern auch für die folgenden möglich. Weitere Rahmen werden entsprechend durch neue ersetzt.

Betätigt der Benutzer bei Darstellung des TCI wie in der obigen Abbildung die sekundäre Maustaste auf der TokenComponent *Diabetes mellitus*, so wird, wie aus der folgenden Abbildung ersichtlich, der zweite Rahmen durch zwei neue ersetzt. Mit den Angaben in den zweiten Rahmen kann der Typ des Diabetes mellitus näher bestimmt werden.

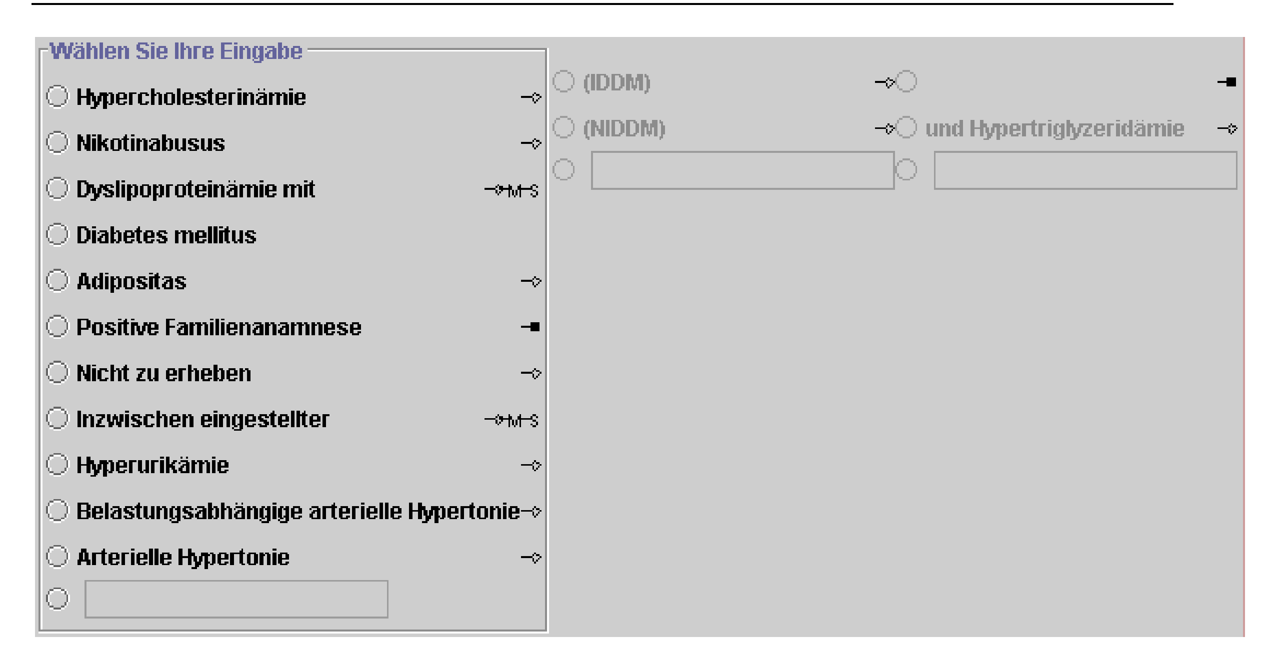


Abbildung 82 TCI-Vorschau

In der Abbildung 82 TCI-Vorschau sind jeweils hinter den TokenComponents Symbole angegeben, die nähere Informationen über den Zustand geben, der nach Betätigung dieser TokenComponent erreicht wird (siehe Kapitel 11.5.2.4.3.3.1 "Anzeige Nachfolger-Symbol").

11.5.6.2.5 Der Vorgang der Texterfassung mit dem TCI

Um einen kompletten Befund mit dem TCI zu erfassen, muß der Benutzer lediglich eine Reihe von TokenComponents mit der Beschriftung anwählen, die in dem fertigen Text stehen soll. Die Beschriftung einer angewählten TokenComponent wird in das unten stehende Textfeld eingetragen. Die Frequenz der Tokens, die die TokenComponent repräsentiert, werden inkrementiert. Der Rahmen, in dem sich die TokenComponent befand, verschwindet, und es wird die nächste Auswahl von TokenComponents in einem neuen Rahmen angezeigt. Diese Auswahl steht für den Zustand, der von der angewählten TokenComponent aus über die Tokens erreicht wird. Ist ein solcher Zustand ein MetaZustand, so wird dieser in das Feld für den MetaTitel eingetragen. In das Textfeld wird er unterstrichen als Überschrift für den kommenden Abschnitt eingetragen.

Der Benutzer fährt solange mit der Anwahl von TokenComponents fort, bis ein TerminalZustand erreicht wird.

Für einen vollständigen Text müssen alle MetaAbschnitte besucht worden sein, denn sonst würde es bedeuten, daß zu einzelnen Abschnitten nichts eingegeben worden ist, was insbesondere für einen vollständigen Befund nicht erwünscht ist, da alle Abschnitte für den Text essentielle Fakten beinhalten. Es wird also nun überprüft, ob alle MetaZustände während des Vorganges der Texterfassung besucht worden sind (siehe

auch Kapitel 11.5.2.4.3.3.3 "Mehrfaches Besuchen von Metazuständen"). Wurde festgestellt, daß ein MetaZustand noch nicht besucht worden ist (dadurch, daß er nicht auf dem vom Benutzer eingeschlagenen Weg lag), so wird dieser als nächster zur Auswahl präsentiert, damit der Benutzer auch zu diesem Abschnitt Angaben machen kann. Sind alle MetaZustände besucht worden, und die zuletzt angewählte TokenComponent führt zu einem TerminalZustand, so ist der Vorgang der Texterfassung beendet. Der letzte Rahmen wird aus dem Eingabefeld entfernt, und es erscheint ein entsprechender Benachrichtigungsdialog.

Wurde dieser durch Betätigung des *OK*-Knopfes bestätigt, so wird das Befundfenster geöffnet (siehe Kapitel 11.5.7 "Das Befund-Fenster").

Ist bei den Optionen für das TCI angegeben worden, daß ein Protokoll des Erfassungsvorganges erstellt werden soll (siehe Kapitel 11.5.6.5 "Das Protokoll-Fenster"), so wird dieses nun angezeigt.

11.5.6.3 Das Textfeld

Durch Betätigung der TokenComponents in der gewünschten Reihenfolge erzeugt der Benutzer einen neuen Text. Dieser Text ergibt sich aus der Konkatenation der Beschriftungen der TokenComponents. Zur besseren Kontrolle des bisher erzeugten Textes wird dieser in das Textfeld eingetragen. Das Erscheinungsbild des Textes ist so, wie er auch später in dem Befundfenster (siehe Kapitel 11.5.7 "Das Befund-Fenster") erscheint. Durch evtl. in den gelernten Texten vorhandene Steuerzeichen erfolgt eine gewisse Formatierung des Textes.

Ist eine MetaStruktur für die Texte erwünscht, so sind die Titel der Abschnitte durch eine doppelte Unterstreichung hervorgehoben. Wie in der Abbildung 81 TokenComponent-Interface Fenster gezeigt, ist der MetaTitel *Koronare Risikofaktoren* unterstrichen. Die Länge der Unterstreichung ist proportional zu der Länge des MetaTitels. Gemäß der Vorgabe der Option aus Kapitel 11.5.2.4.3.3.4 "Mehrfache Anzeige von Metatiteln" werden die MetaTitel entweder nur einmal oder bei einem weiteren Besuch auch mehrfach eingefügt.

11.5.6.4 Die Werkzeugleiste

11.5.6.4.1 Legende

Durch Betätigung dieses Knopfes öffnet sich ein Dialog, welcher Auskunft über die einzelnen Symbole gibt, die neben den TokenComponents stehen können.



Abbildung 83 Legende für das TokenComponent-Interface

Soll die Anzeige der Symbole unterdrückt werden, so kann dies durch Betätigung der entsprechenden Option geschehen (siehe Kapitel 11.5.2.4.3.3.1 "Anzeige Nachfolger-Symbol").

11.5.6.4.2 Erstellen

Wurde ein Texterfassungsvorgang abgeschlossen oder ein neuer Graph geöffnet, so ist es notwendig, das TCI neu zu erstellen, um neue Eingaben in das TCI zu ermöglichen. Durch Betätigung dieses Knopfes wird der erste Rahmen mit den TokenComponents für den StartZustand des DFA erstellt.

11.5.6.4.3 Beenden

Der *Beenden*-Knopf ist zur weiteren Beschleunigung des Texterfassungsvorganges. Wird er betätigt, so vervollständigt *mAGENTa* automatisch den Text mit den wahrscheinlichsten Tokens bis zum nächsten MetaZustand. Dies geschieht durch Auswahl der Tokens mit der jeweils höchsten Frequenz pro Zustand, an dem der Vorgang gerade weitergeführt wird. Diese Vorgehen gleicht der Auswahl der jeweils ersten TokenComponent in einem Rahmen, wenn die Sortierung nach Frequenz eingeschaltet ist (siehe Kapitel 11.5.2.4.3.3.5 "Nach Frequenz").

11.5.6.4.4 Zurück

Nicht jede Entscheidung zur Anwahl einer TokenComponent stellt sich im Nachhinein als richtig und gewünscht dar. *mAGENTa* bietet hierfür eine Rückgängig-Funktion („Undo“) für alle Betätigungen von TokenComponents. Ist irrtümlicherweise eine TokenComponent angewählt worden, die gar nicht beabsichtigt war, so kann mittels des *Zurück*-Knopfes dieser Vorgang zurückgenommen werden. Die Mächtigkeit dieser Funktion erlaubt es, sämtliche Eingaben in das TCI zurückzunehmen, bis wieder der Rahmen für den StartZustand dargestellt wird. Sogar die Eingaben in EmptyComponents lassen sich rückgängig machen, auch nachdem sie gelernt worden sind.

Die Frequenzen der einzelnen Tokens, die durch Betätigung einer TokenComponent erhöht worden sind, werden wieder entsprechend dekrementiert.

11.5.6.4.5 Weiter

Der *Weiter*-Knopf wird betätigt, um das Ende der Eingabe in eine EmptyComponent zu signalisieren.

Wurde der Text eingetragen und anschließend dieser Knopf betätigt, so wird eine Folge von Zuständen erzeugt, die die eingegebenen Worte jeweils zur Überführung in den nächsten Zustand nutzen. Der nachfolgende Zustand der EmptyComponent ist der nächste MetaZustand oder, wenn keiner erreichbar ist, der TerminalZustand. Diente ein TokenButton als „Vorbild“ für die EmptyComponent, so führen beide zu dem nachfolgenden Zustand des TokenButtons.

11.5.6.4.6 Merge

Ist in dem Dialog zur Festlegung der Merging-Parameter (siehe Kapitel 11.5.2.4.3.1.1 "Festlegen der Merging-Parameter...") die Option *Sofortige Anwendung der Regeln* selektiert, so wird direkt nach Eingabe in eine EmptyComponent der hinzugefügte Text gelernt. Der *Merge*-Knopf ist nicht aktivierbar.

Wenn die Option allerdings nicht selektiert ist, so werden die Einträge zwar dem DFA hinzugefügt, aber noch nicht gelernt. Dies erkennt *mAGENTa* und der *Merge*-Knopf ist aktivierbar. Der Benutzer kann nun jederzeit den Knopf betätigen, um den Lernprozeß zu starten. Es ist somit möglich den Lernprozeß erst nach mehreren Eingaben in verschiedene EmptyComponents, oder erst nach einigen Texterfassungsvorgängen zu starten, ganz wie es der Benutzer wünscht.

Vgl. auch das Kapitel 11.5.5.2.5 "Merge" aus dem Wörterbuch.

11.5.6.5 Das Protokoll-Fenster

Ist die Option, daß der Texterfassungsvorgang protokolliert werden soll (siehe Kapitel 11.5.2.4.3.3.6 "Protokoll der TC-Positionen") selektiert, so erscheint am Ende des Vorganges das TCI-Protokoll in einem eigenen Fenster. Das Protokoll gibt einerseits Auskunft über die Geschwindigkeit, mit der der Text mittels des TCI erfaßt worden ist, andererseits wird festgehalten, wie häufig die TokenComponents an einer bestimmten Position betätigt worden sind. Neben den absoluten Zahlen erfolgt auch eine prozentuale Gewichtung, die angibt, wie gut die Sortierung der TokenComponents helfen kann, neue Texte zu erfassen. Es sollten bei einem neu erstellten Text die in einem Rahmen weiter oben stehenden TokenComponents häufiger betätigt werden als diejenigen im unteren Bereich, da die oberen gemäß ihren Frequenzen auch in der Vergangenheit häufiger vorkamen.

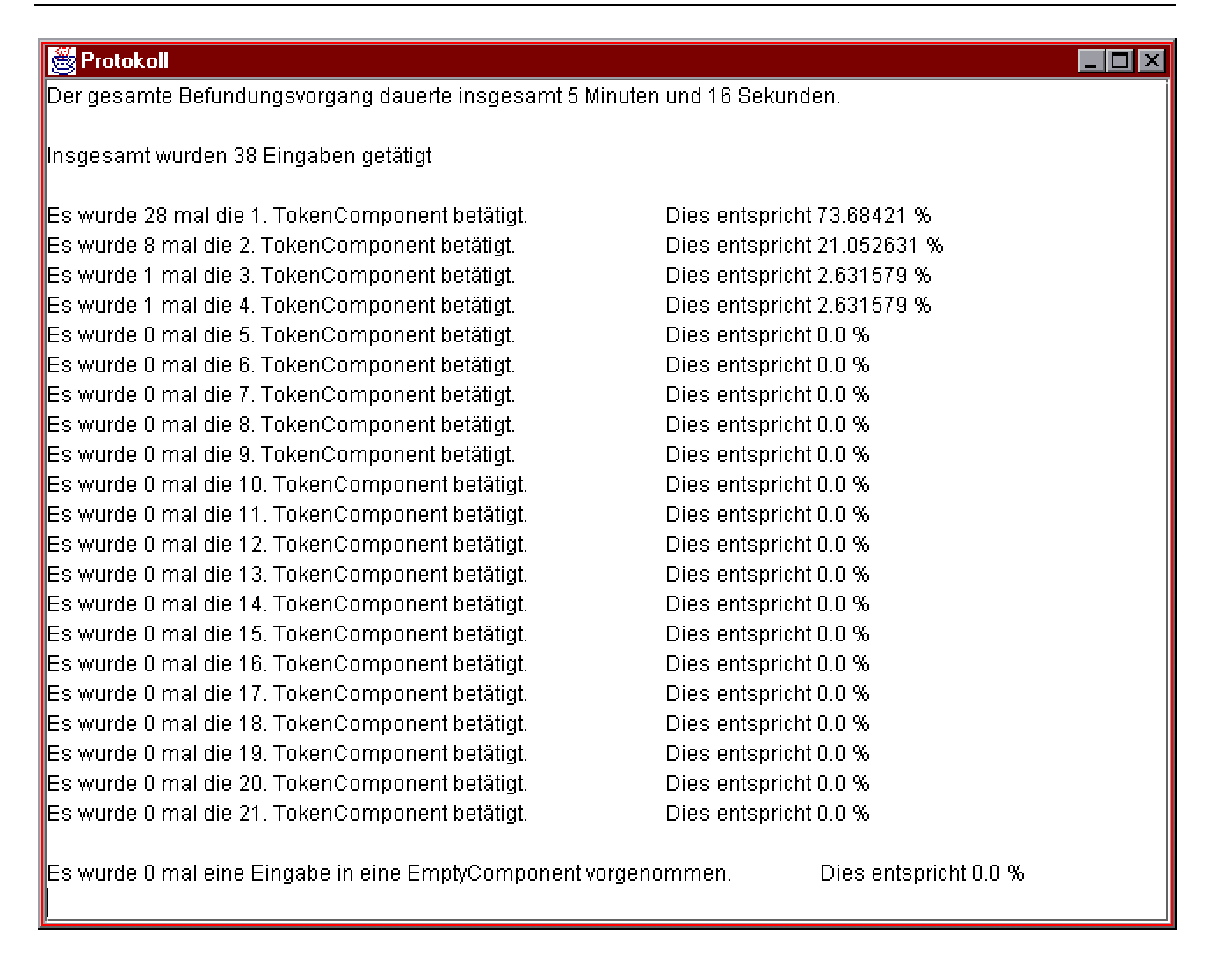
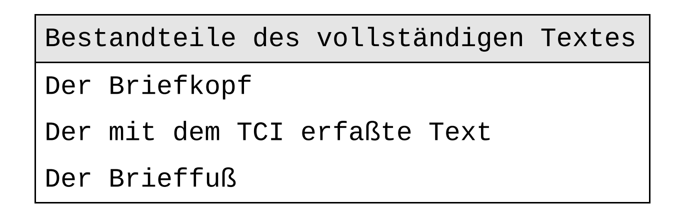


Abbildung 84 TCI-Protokoll

Der letzte Eintrag für eine TokenComponent (in der obigen Abbildung 21) ist auch gleichzeitig der maximale Verzweigungsgrad des DFA.

11.5.7 Das Befund-Fenster

Ist die Texterfassung mittels des TCI abgeschlossen, so erscheint in einem separaten Fenster der vollständige Text. Dieser vollständige Text setzt sich aus drei Teilen zusammen:



Bestandteile des vollständigen Textes
Der Briefkopf
Der mit dem TCI erfaßte Text
Der Brieffuß

Tabelle 28 Bestandteile des vollständigen Textes

Zur Information über den Briefkopf und -fuß siehe Kapitel 11.5.2.4.3.3.8 "Briefkopf..." bzw. 11.5.2.4.3.3.9 "Brieffuß...". Der erfaßte Text wird direkt aus dem Textfeld des TCI übernommen (siehe Kapitel 11.5.6.3 "Das Textfeld").

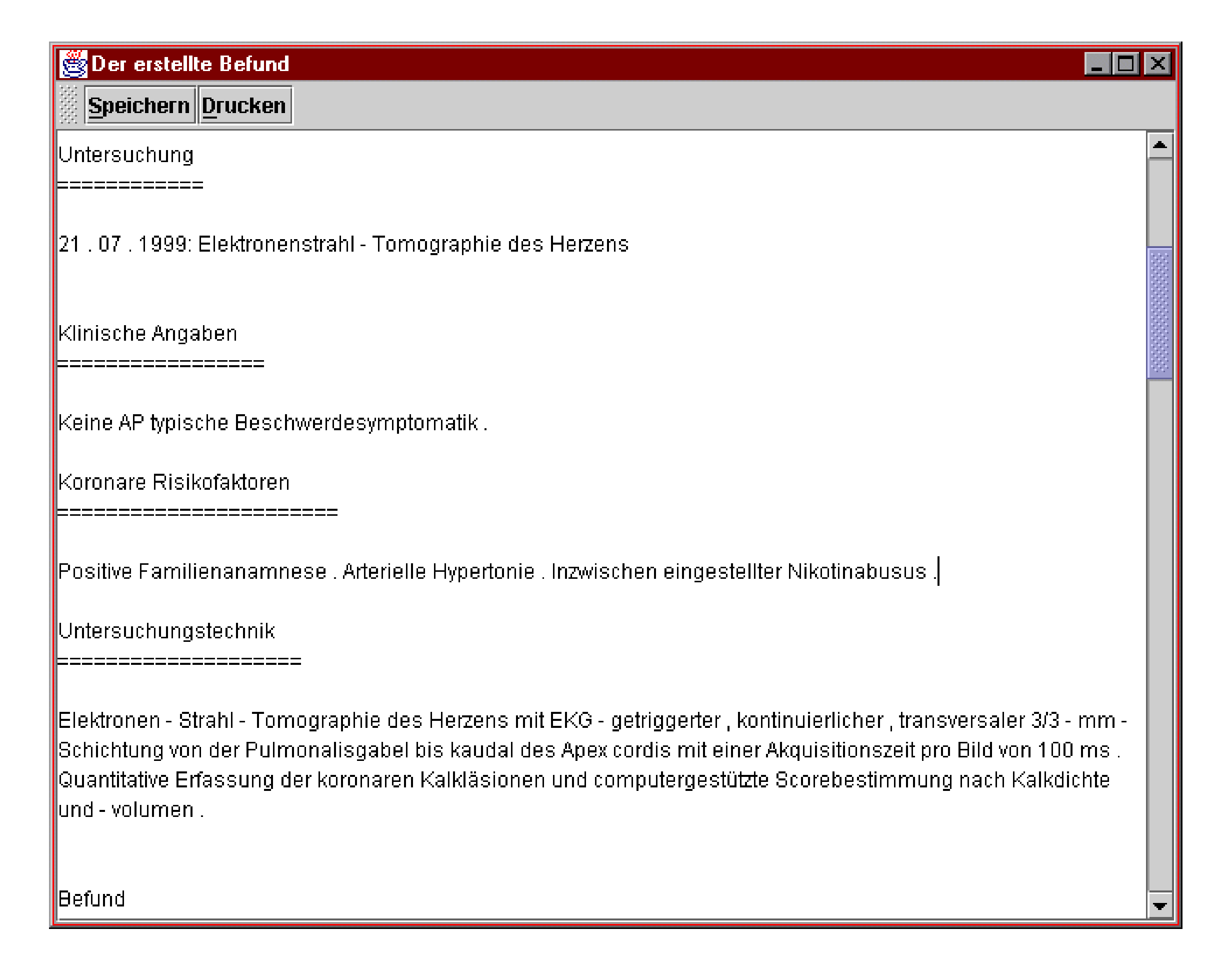


Abbildung 85 Befund-Fenster

11.5.7.1 Das Textfeld

In der obigen Abbildung ist ein Ausschnitt aus solch einem vollständigen Text zu sehen. Durch den Rollbalken am rechten Rand des Fensters kann der Rest des Textes eingesehen werden. Falls der Benutzer es wünscht, kann der Text in dem Textfeld noch weiter editiert werden, um weitere Eintragungen oder Löschungen von Textpassagen zu tätigen. Dies sollte natürlich nicht mehr notwendig sein, da alle Eingaben über das TCI erfolgen können.

Ist die Option aktiviert, daß während des Lernprozesses eingelesene Zahlen durch das vorgegebene Token „XXX“ ersetzt werden sollen (siehe Kapitel 11.5.2.4.3.4.4 "Ersetzen von Zahlen"), so können hier dieser und andere Platzhalter manuell durch den gewünschten Eintrag ersetzt werden.

11.5.7.2 Die Werkzeugleiste

Die Werkzeugleiste bietet für die Texterfassung grundlegende Funktionen: Speichern und Drucken. Beide Funktionen können aus Sicherheitsgründen nur lokal wirksam werden (siehe Kapitel 11.5.2.4.3.4.7 "Internet-Verbindung vorhanden"). Die Knöpfe sind nicht aktivierbar, wenn die Internet-Verbindung besteht.

11.5.7.2.1 Speichern

Wird der *Speichern*-Knopf betätigt, so öffnet sich ein Dateiauswahldialog, in dem angegeben werden kann, unter welchem Dateinamen der erstellte Text abgespeichert werden soll (siehe 11.5.2.4.1.5 Speichern unter...). Die Datei wird in einem normalen Textformat gespeichert, so daß sie einfach zur Weiterverarbeitung von anderen Anwendungen importiert werden kann.

11.5.7.2.2 Drucken

Der Text kann aber auch direkt auf einem angeschlossenen Drucker ausgegeben werden. *mAGENTa* nutzt hierzu jeweils die Schnittstellen des aktuellen Betriebssystems, um die Druckoperation durchzuführen.

11.6 *mAGENTa*Script

*mAGENTa*Script ist ein zusätzliches Programm, welches es erlaubt, den Lernprozeß von vielen Beispieltexten zu automatisieren. Es ist eine Java-Applikation, die ohne graphische Oberfläche auskommt, und per Kommandozeile durch Aufruf des Java-Interpreters gestartet wird.

Als Parameter benötigt *mAGENTa*Script zwei Dateinamen. Die erste Datei muß eine Liste von weiteren Dateien enthalten, in denen jeweils die Texte stehen, die gelernt werden sollen. Die zweite Datei enthält in kodierter Form pro Zeile jeweils Parametereinstellungen für den Lernprozeß.

*mAGENTa*Script ließt nun für eine Parametrisierung alle Texte ein, lernt sie und speichert den gelernten DFA inklusive des Protokolles des Lernvorganges in ein eigenes Verzeichnis ab. Dies wird für jede Zeile der Datei mit Parametereinstellungen durchgeführt.

*mAGENTa*Script hat den Vorteil, daß es unbeaufsichtigt eine Fülle von Lernvorgängen ausführen kann. Durch Verzicht auf eine graphische Oberfläche ist es möglich das Programm auf entfernten Rechnern („remote“), insbesondere auf Workstations, ausführen zu lassen, um so den Arbeitsplatzrechner nicht mit dem Lernprozeß zu belasten.

11.7 generateIndex

generateIndex ist eine Java-Application, die HTML-Indexdateien zu Verzeichnissen generieren kann. Für ein oder mehrere Verzeichnisse werden rekursiv Indexdateien generiert, die als Inhalt Verweise auf die einzelnen Dateien in dem Verzeichnis haben.

Es wurden hiermit Indexdateien für die in Tabelle 14 Verzeichnisse auf dem Server angegebenen Verzeichnisse erstellt, damit beim Öffnen von Dateien über das Netzwerk eine einfachere Auswahl der Dateien ermöglicht wird (siehe Kapitel 11.5.2.3.2.2 "Öffne externe Datei aus dem Netzwerk").

In Abbildung 55 Öffne externe Datei mittels eines Internet-Browsers ist im Hintergrund eine Indexdatei für ein Verzeichnis mit EBT-Befunden zu sehen.

III Auswertung

In diesem Kapitel werden die Testergebnisse aus drei unterschiedlichen radiologischen Domänen (CT HWS, CT LWS und MRT LWS) im Umfang von insgesamt ca. 250 Befunden betrachtet.

Ziel der nachfolgenden Testreihen war es zum einen den Nachweis der Domänenunabhängigkeit des Systems zu erbringen und zum anderen das subjektive Kriterium der Benutzerfreundlichkeit der jeweils erzeugten Oberflächen im Kontext der radiologischen Befundung quantitativ zu erfassen.

Dabei konzentrieren sich die folgenden Untersuchungen in Ergänzung zu den in der Einleitung der Diplomarbeit abstrakt formulierten Anforderungen an das System wie z. B. Kategorisierung, Vereinheitlichung, Fehlervermeidung, kooperative Nutzung etc. primär auf formale Kriterien, die eine objektive Bewertung der gelernten Automaten in Hinblick auf die Benutzerfreundlichkeit der entsprechenden Oberflächen und deren Verwendbarkeit im praktischen Einsatz ermöglichen.

Bei der Erstellung der Testreihen wurde von der Vielzahl der möglichen Parametereinstellungen des Systems abstrahiert und der Fokus der Betrachtung auf die Auswirkungen bestimmter Kombinationen von Mergingregeln gesetzt, die das jeweilige Lernergebnis entscheidend beeinflussen.

12 Testergebnisse

12.1 Einleitung

Wie bereits erwähnt werden im folgenden die Auswirkungen bestimmter Regelmengen auf das Lernergebnis, also insbesondere auf die Benutzerfreundlichkeit der jeweiligen Oberflächen untersucht. Dabei werden insgesamt acht Regelkombinationen basierend auf den Ansätzen von Angluin, Schlimmer und Hermens und einer Erweiterung des letzten Ansatzes untersucht.[12]

Ein wichtiges Kriterium zur formalen Evaluation des Lernerfolges ist das bereits in Kapitel 5.2.6.1 beschriebene Paradigma der Identifikation im Grenzwert. Eine Methode den formalen Lernerfolg im konkreten Fall zu bewerten ist es zu überprüfen, ob der entstandene DFA übergeneralisiert, d. h. ob er ungrammatikalische Sätze erzeugt.

Im einzelnen werden für jede Domäne neben dem entsprechenden Präfixbaum die Mergingergebnisse nach Anwendung der Regelmengen von Angluin, Schlimmer und Hermens sowie einer modifizierten Version der letzteren Regelmenge jeweils mit den k-Vorgängerwerten 0 und 2 anhand diverser Kriterien bewertet. Dabei soll der Fokus der Betrachtung jeweils auf diejenigen Regelkombinationen gelegt werden, die das jeweilige Kriterium besonders gut bzw. besonders schlecht erfüllen.

Der k-Vorgängerwert 2 wurde im übrigen gewählt, da im Vorfeld der Testreihen auf der Grundlage des in Kapitel 8.3.3.2 beschriebenen Verfahrens von [Berwick et al. 1987] die 2-Reversibilität der betrachteten Domänen gezeigt wurde.

Die beschriebenen Regelkombinationen werden im weiteren Verlauf der Auswertung folgendermaßen kodiert: Nach jeder Regelangabe (2a, 2b, 3a, 3aE und 3b) folgt jeweils der entsprechende k-Vorgängerwert (z. B. bedeutet 12a22b23a13b1 einerseits, daß sich die Regelmenge aus den Regeln 1, 2a, 2b, 3a und 3b zusammensetzt und andererseits, daß die Regeln 2a und 2b einen k-Vorgängerwert von 2, sowie die Regeln 3a und 3b einen k-Vorgängerwert von 1 besitzen).

Das Ausrufezeichen im Zusammenhang mit der Regelmenge 12a2b bedeutet, daß in einem Vorverarbeitungsschritt sämtliche Zahlenwerte durch den String „###“ ersetzt werden (für weitere Erläuterungen siehe Kapitel 13.5.2.4.3.4.4).

---

[12] Eine ausführliche Diskussion des Mergingprozesses und der verwendeten Mergingregeln findet sich in Kapitel 8.3

12.2 Formale Bewertungskriterien

12.2.1 Prozentualer Zuwachs der Wortanzahl

Abbildung 86 zeigt, daß unabhängig von den jeweiligen Domänen und den gewählten Mergingparametern im Verlauf des Lernvorganges der prozentuale Anteil neuer Worte stetig abnimmt und somit die Menge der vom jeweiligen DFA neu erzeugten Sätze gegen Null konvergiert.

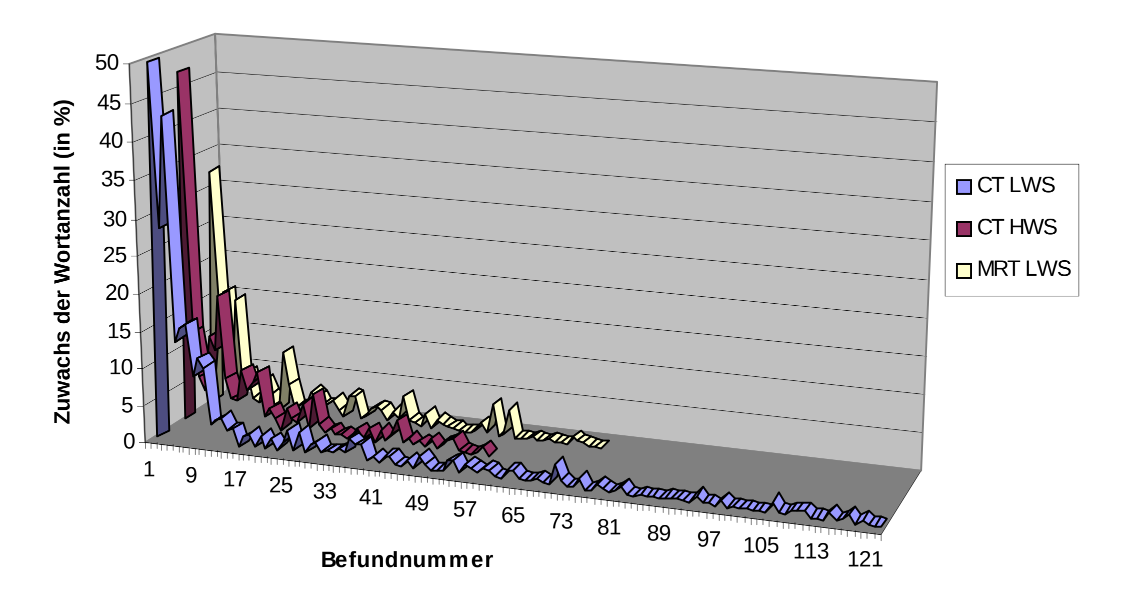


Abbildung 86 prozentualerZuwachs der Wortanzahl

12.2.2 Komprimierungsgrad des DFA

Der Komprimierungsgrad eines endlichen Automaten (bezüglich des jeweiligen Präfixbaumes) kann als ein Indikator für die Qualität des DFA gewertet werden.

Generell ist ein hoher Komprimierungsgrad ein Indiz für einen entsprechend hohen Zyklenanteil (siehe Kapitel 12.2.3) und hat in der Regel eine Verringerung der Benutzerfreundlichkeit der entsprechenden Oberflächen zur Folge. Andererseits zeigen die nachfolgenden Testergebnisse, daß ein niedriger Komprimierungsgrad von ca. 30-40%, wie im Fall der Regelmenge 12a02b0, kein hinreichendes Kriterium ist um eine subjektive Benutzerfreundlichkeit des Systems zu garantieren.

Des weiteren zeigt Abbildung 87, daß die ursprüngliche (12a02b03a13b1) und die erweiterte (12a02b03aE13b1) Regelmenge von Schlimmer und Hermens mit einem k-Vorgängerwert von 0 einen Komprimierungsgrad von bis zu 94% aufweisen, wohingegen der Komprimierungsgrad nach Anwendung der Regelmenge von Angluin mit einem k-Vorgängerwert von 2 (12a22b2 und 12a22b2!) mit 25-50% minimal bezüglich der betrachteten Regelmengen sind.

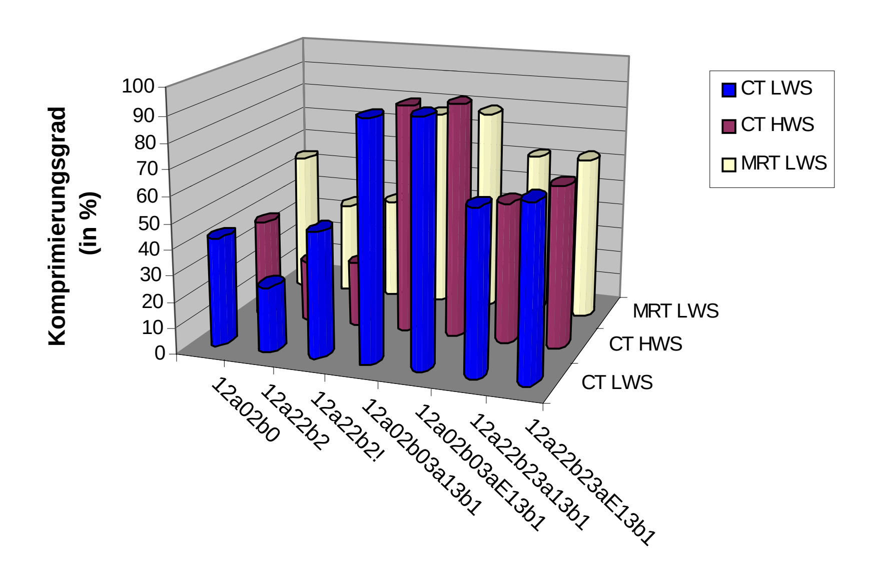


Abbildung 87 Komprimierungsgrad bezüglich der Präfixbäume

12.2.3 Anteil der Zyklen des DFA

Wie bereits erwähnt, wurde der Anteil der Zyklen eines Automaten als Maß für den Grad der Übergeneralisierung gewählt, da eine Quantifizierung der Übergeneralisierungen und somit eine direkte Vergleichbarkeit zweier Automaten bezüglich dieses Kriteriums nicht ohne weiteres realisierbar ist.

Der Zyklenanteil wirkt sich dabei unmittelbar auf die Benutzerfreundlichkeit des Systems aus, da dieser zum einen den Befundungsprozeß erschwert und zum anderen der Trainingseffekt durch die Vielzahl der ungrammatikalischen Auswahlmöglichkeiten gefährdet ist.

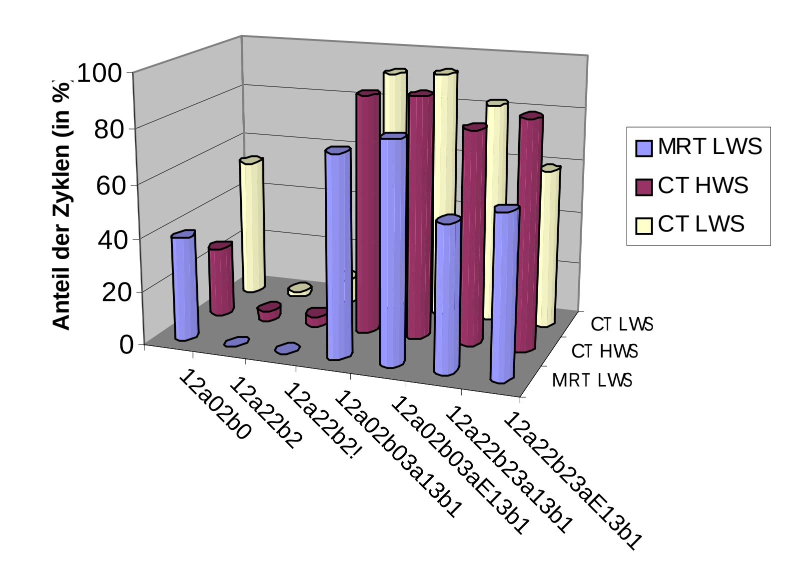


Abbildung 88 prozentualer Zyklenanteil

Aus der Abbildung 88 geht deutlich hervor, daß der Anteil der Zyklen im Fall der Regelmenge von Angluin mit einem k-Vorgängerwert von 2 (12a22b2 und 12a22b2!) minimal ist im Vergleich zu den restlichen Regelkombinationen. Die Anwendung der ursprünglichen und der modifizierten Regelmengen von Schlimmer und Hermens insbesondere mit einem k-Vorgängerwert von 0 (12a02b03a13b1 bzw. 12a02b03aE13b1) ergibt dabei einen teilweise erheblichen Zyklenanteil.



Abbildung 89 Benutzeroberfläche eines übergeneralisierenden Automaten

Abbildung 89 zeigt einen Ausschnitt aus einer entsprechenden Oberfläche mit hohem Zyklenanteil. Es ist unschwer zu erkennen, daß ein effektiver Befundungsvorgang mit einer derartigen Benutzeroberfläche nicht zu realisieren ist.

Das vorliegende Ergebnis dieser Testreihe deckt sich dabei sowohl mit den subjektiven Bewertungen der jeweiligen Oberflächen als auch mit den theoretischen Erwartungen bezüglich der Auswirkungen eines hohen Zyklenanteils.

12.2.4 Durchschnittlicher Fan-Out

Der durchschnittliche Fan-Out[13] der erzeugten DFA ist ein weiterer Faktor, der sich unmittelbar auf die Benutzerfreundlichkeit des Systems auswirkt. Der Fan-Out eines Zustandes korreliert dabei mit der Anzahl der TokenComponents des entsprechenden Zustandspanels, d. h. ein hoher Fan-Out hat eine umfangreiche und in der Regel unübersichtliche Benutzeroberfläche zur Folge.

Auch in diesem Punkt zeigen die Regelmengen von Schlimmer und Hermens mit einem k-Vorgängerwert von 0 das ungünstigste Verhalten. Die Ergebnisse der restlichen Regelmengen unterscheiden sich hingegegen kaum voneinander, wobei in diesem Zusammenhang ein leichter Vorteil bei der Regelmenge von Angluin mit einem k-Vorgängerwert von 2 zu verzeichnen ist.

Auffallend sind auch die maximalen Fan-Out-Werte innerhalb der einzelnen Domänen. So übersteigt der Maximalwert der Domäne CT LWS den entsprechenden Wert der Domäne MRT LWS um das dreifache. Diese Tatsache ist auf die insgesamt inhomogene Struktur der Domäne zurückzuführen.

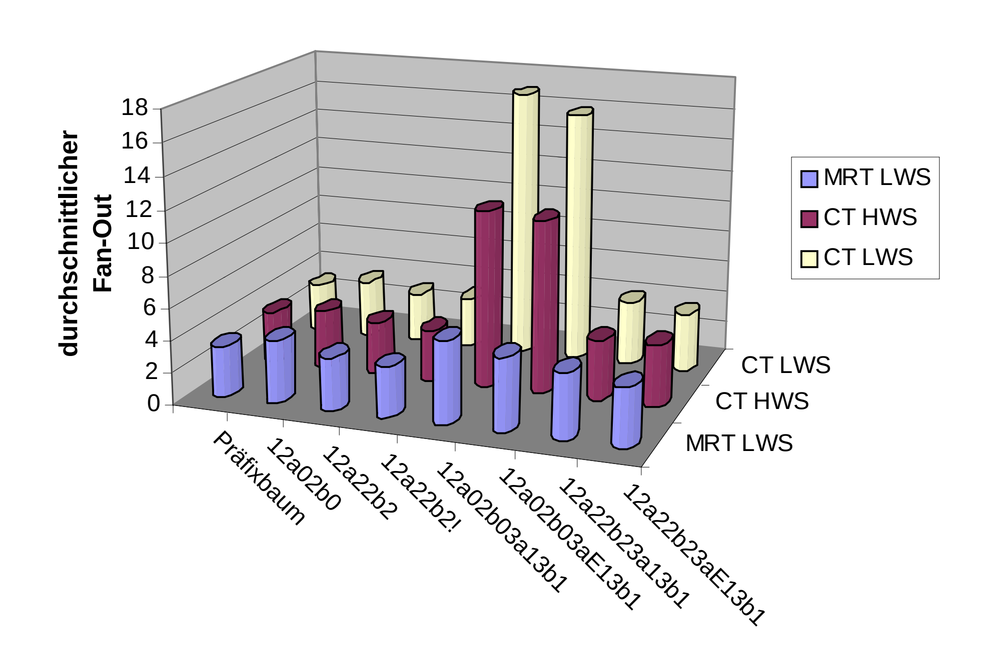


Abbildung 90 durchschnittlicher Fan-Out

---

[13] Da ca. 98% aller Zustände exakt einen Zustandsnachfolger besitzen (Fan-Out=1) werden bei der Ermittlung des durchschnittlichen Fan-Outs ausschließlich Zustände mit mehr als einem Zustandsnachfolger betrachtet, um eine bessere Vergleichbarkeit der einzelnen Regelmengen zu gewährleisten.

12.2.5 Anzahl der Eingaben je Befundungsvorgang

Die Anzahl der Eingaben im Rahmen eines einzelnen Befundungsvorganges entspricht der Zahl der unterschiedlichen TokenComponents die der Benutzer anwählen muß, um vom Startzustand des DFA zum Terminalzustand zu gelangen und ist somit ein Faktor, der die Dauer dieses Vorganges entscheidend beeinflußt. Die Zahl der Eingaben bezieht sich dabei jeweils auf die Tokensequenzen mit dem höchsten Frequenzwert innerhalb eines Zustandspanels. Vereinfacht gesagt wird also die Länge des wahrscheinlichsten Weges durch das TokenComponentInterface gemessen.

Es kann festgehalten werden, daß die durchschnittliche Länge der einzelnen Tokensequenzen umgekehrt proportional zur Zahl der Eingaben ist.

Die Praxis zeigt hier, daß sowohl viele Eingaben mit jeweils kurzen Tokensequenzen, als auch wenige Eingaben mit entsprechend langen Tokensequenzen zu einer subjektiv schlechten Bewertung im Sinne der Benutzerfreundlichkeit führen. Im ersten Fall ist der Zeitaufwand für einen durchschnittlichen Befundungsvorgang relativ groß, im zweiten Fall ist die Wahrscheinlichkeit hoch, daß die vom Benutzer ausgewählten Sequenzen unerwünschte Textfragmente enthalten.

Ein optimales Ergebnis im Hinblick auf die Benutzerfreundlichkeit des Systems wird in der Regel durch einen DFA erreicht, bei dem sich die Anzahl der Eingaben zwischen diesen beiden Extremen bewegt.

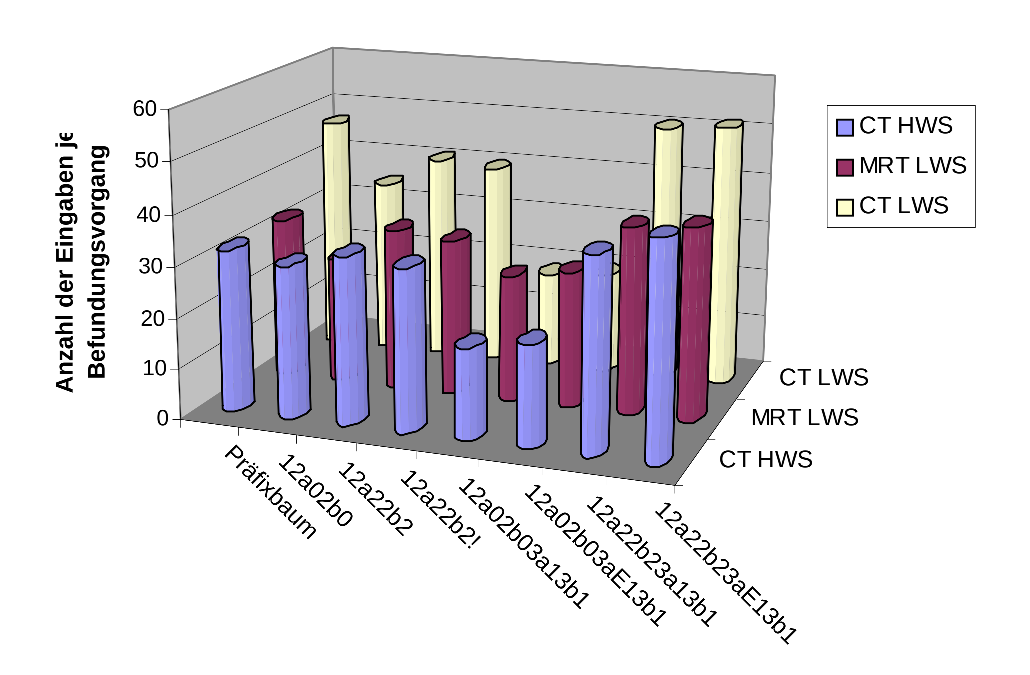


Abbildung 91 Anzahl der Eingaben je Befundungsvorgang

Neben dem oben beschriebenen Zusammenhang zwischen der Anzahl der Eingaben innerhalb eines Befundungsvorganges und der durchschnittlichen Länge der einzelnen Tokensequenzen wird noch ein weiterer Zusammenhang deutlich. Gemeint ist die Relation zwischen dem durchschnittlichen Fan-Out eines Zustandes und der Länge der einzelnen Tokensequenzen im entsprechenden Zustandspanel. Je höher der Fan-Out desto geringer die Länge der Tokensequenzen, bzw. vice versa. Auch hier läge der optimale Wert im Sinne der Benutzerfreundlichkeit zwischen diesen beiden Extremen.

Abbildung 91 zeigt, daß bei der Regelmenge von Angluin mit einem k-Vorgängerwert von 2 ein durchschnittlicher und somit optimaler Wert in Bezug auf die Anzahl der Eingaben erreicht wird. Der höchsten Wert wird bei der Regelmenge von Schlimmer und Hermens mit einem k-Vorgängerwert von 2, der geringste Wert mit einem Vorgängerwert von 0 angenommen..

12.2.6 Fan-Out der Meta-Zustände

Aus den nächsten Abbildungen kann man folgenden Erkenntnisse ableiten. Zum einen führen Inhomogenitäten einzelner Meta-Abschnitte (*Segment C3/4*, *Segment C4/5, Segment C5/6 und Segment C6/7* in der Domäne CT HWS, *Segment L3/4*, *Segment L4/5*, *Segment L5/S1* in der Domäne CT LWS bzw. *Beurteilung*, *Segment L4/5*, *Segment L5/S1* in der Domäne MRT LWS) bezüglich der zugrundeliegenden Syntax im Zusammenhang mit den Regelmengen 12a02b03a13b1 bzw. 12a02b03aE13b1 zu hohen maximalen Fan-Out-Werten von bis zu 150 (CT HWS), bzw. sogar bis zu 280 (CT LWS). Demgegenüber weisen homogenere Abschnitte (wie z. B. *Fragestellung* oder *Untersuchungstechnik*) unabhängig von den jeweiligen Domänen einen geringen Fan-Out auf.

Zum anderen wird bezüglich der Regelmenge 12a2b2! deutlich, daß die Ersetzung von Zahlwerten im Vorfeld des eigentlichen Mergingvorganges eine wesentlich bessere Generalisierung in Abschnitten mit einem naturgemäß hohen Fan-Out (z. B. bei Untersuchungsdaten im Abschnitt *Untersuchung*) zur Folge hat.

Diese Ergebnisse bestätigen die in Kapitel 12.2.4 erzielten Erkenntnisse bezüglich der Auswirkungen einzelner Regelmengen auf den Fan-Out des entstehenden Automaten.

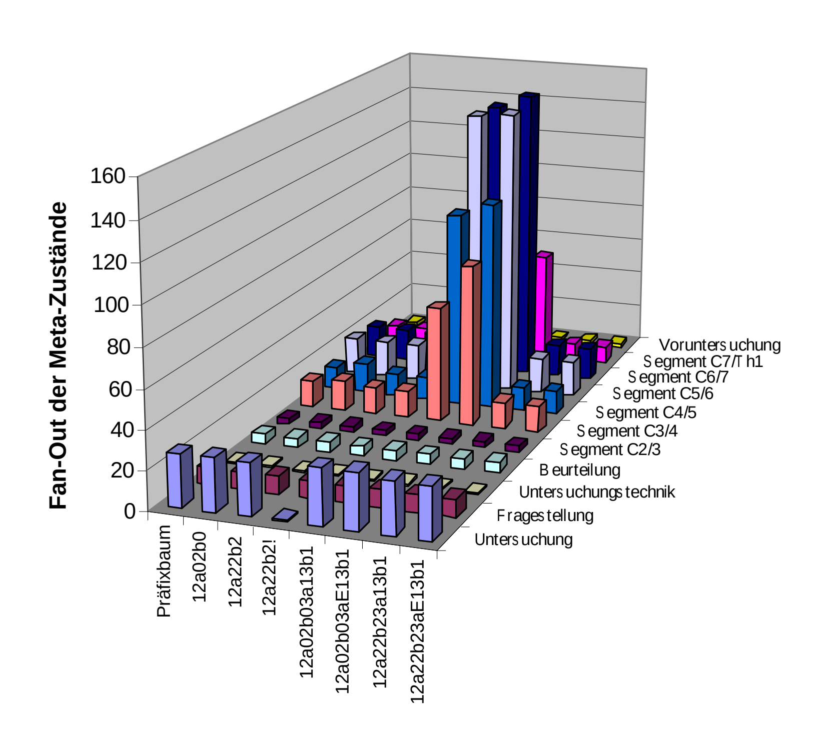


Abbildung 92 Fan-Out der Meta-Zustände (CT HWS)

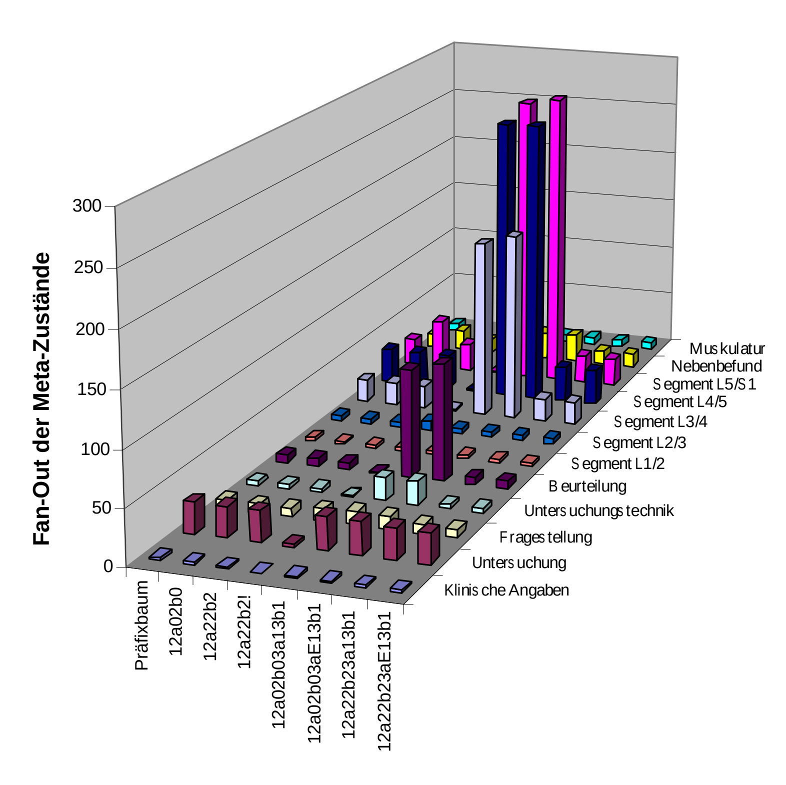


Abbildung 93 Fan-Out der Meta-Zustände (CT LWS)

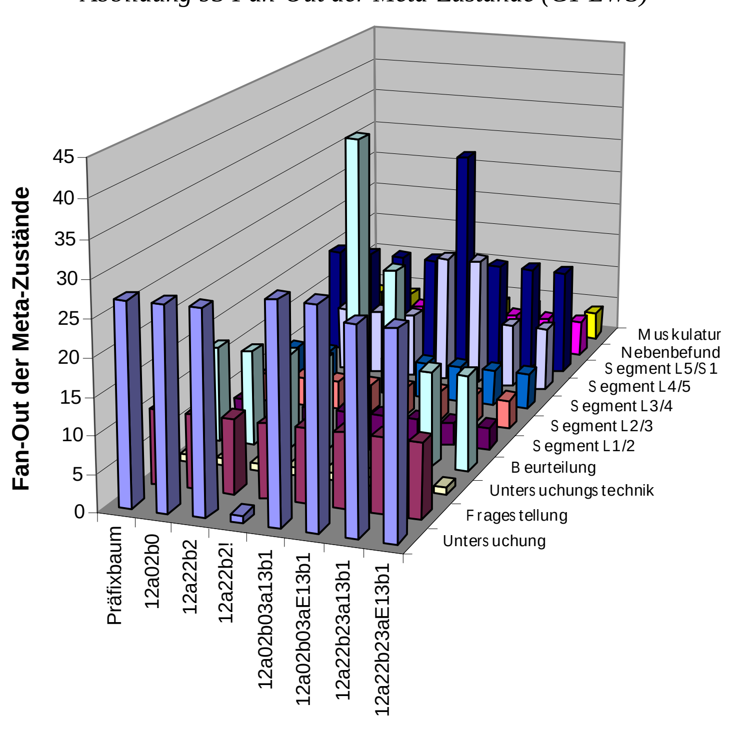


Abbildung 94 Fan-Out der Meta-Zustände (MRT LWS)

12.3 Bewertung der Testergebnisse

Es kann abschließend festgehalten werden, daß ein direkter Zusammenhang zwischen der objektiven Bewertung des jeweiligen DFA durch formale Kriterien wie dem Komprimierungsgrad, dem Anteil der Zyklen, dem durchschnittlichen Fan-Out und der Anzahl der Einaben je Befundungsvorgang auf der einen Seite und der subjektiven Bewertung der entsprechenden Benutzeroberflächen auf der anderen Seite besteht.

Die Automaten, bei denen die Benutzeroberflächen subjektiv am besten bewertet wurden wiesen je nach Domäne einen Zyklenanteil von maximal 9%, einen Komprimierungsgrad zwischen 25 und 48%, eine durchschnittliche Anzahl von ca. 30 Eingaben je Befundungsvorgang, einen durchschnittlichen Fan-Out von 3,1 und den geringsten Grad an Übergeneralisierungen innerhalb der Vergleichsmenge auf. Die obigen Werte wurden mit der Regelmenge 12a22b2! erreicht.

Ein formaler Hinweis auf die Domänenunabhängigkeit des Systems ist, neben dem bereits in Kapitel 12.2.1 beschriebenen Zusammenhang, daß die betrachteten Domänen bezüglich der einzelnen Kriterien vergleichbare Profile auswiesen. Dieses Verhalten ließ sich sowohl für den Komprimierungsgrad als auch den Anteil der Zyklen, den durchschnittlichem Fan-Out und die Anzahl der Einaben je Befundungsvorgang beobachten.

Der Zeitaufwand für einen Befundungsvorgang hängt im Grunde von mehreren Kriterien ab. Zum einen hemmt ein hoher durchschnittlicher Fan-Out diesen Vorgang, da der Arzt unter Umständen die gesamte Liste der Tokensequenzen durchsehen muß um einen gewünschten Eintrag zu finden. Zum anderen erschwert ein hoher Zyklenanteil bzw. eine hohe Anzahl von Eingaben eine effektive Befundung und wirkt sich negativ auf die Geschwindigkeit aus.

13 Abschließende Betrachtung

13.1 Zusammenfassung

Die im ersten Abschnitt formulierten Anforderungen an ein Befundungsunterstützungs-system werden mit dem hier vorgestellten Softwareagenten erfüllt. Im folgenden wird konkret auf einzelne Anforderungen und ihre entsprechende Realisierung im System eingegangen.

zu 1. Kategorisierung:

Für jede Befundkategorie ist es möglich jeweils einen eigenen Agenten zu trainieren. Im Rahmen umfangreicher Testreihen wurde exemplarisch eine Kategorisierung in die Domänen CT HWS, CT LWS und MRT LWS vorgenommen. Eine Einteilung der radiologischen Befundung in weitere Befundkategorien ist denkbar und mit dem vorliegenden System ohne weiteres zu realisieren.

zu 2. Strukturierung:

Mithilfe des in Kapitel 8.3.1 beschriebenen MetaStruktur-Konzeptes und des Aufbau des DFA wird der Arzt im TCI strukturiert durch den Befundungsvorgang geleitet.

zu 3. Vereinheitlichung:

Hierzu dient das Wörterbuch mit dessen Hilfe automatisch Synonyme, Akronyme, Abkürzungen usw. in einer einheitlichen Schreibweise dargestellt werden können. Das Wörterbuch verfügt über die Möglichkeit eine beliebige Folge von Wörtern durch eine andere Folge zu ersetzen (n:m-Ersetzung). Somit kann in gewissem Umfang das Problem einer einheitlichen Terminologie in der radiologischen Befundung durch entsprechend umfassende Wörterbücher gelöst werden.

zu 4. Fehlervermeidung:

Da der Arzt seinen Befund direkt mittels der vom System erzeugten Benutzeroberfläche (durch Anwahl einzelner Radiobuttons) erstellt und diese Oberfläche jeweils aus korrekten Befunddaten gelernt wurde ist die Gefahr inkorrekter Eingaben weitgehend ausgeschlossen. Fehlerhafte Eingaben, die der Arzt zusätzlich im Rahmen des Befundungsvorganges über entsprechende EmptyComponents vornimmt werden Mithilfe des Wörterbuches automatisch korrigiert.

zu 5. Adaptivität:

Das System beobachtet das Verhalten des Benutzers, paßt sich diesem automatisch durch entsprechende Lernalgorithmen an und kann auch interaktiv durch Eingabe von Texten in das TCI ergänzt werden. Falsch geschriebene Wörter können mit Hilfe des Wörterbuches direkt durch korrekte ersetzt werden. Neben der automatischen Anpassung der Systems an den Benutzer hat dieser selbst die Möglichkeit über zahlreiche Parameter das Verhalten des Systems und seiner Komponenten (Tokenizer, Wörterbuch, Lernkomponente und TCI) zu beeinflussen.

zu 6. kooperative Nutzung:

Die Benutzung und das Training des Systems kann von verschiedenen Ärzten getrennt geschehen. Dabei können sowohl die erlernte Struktur (in Form des DFA) als auch das jeweilige Wörterbuch der einzelnen Agenten zusammengefaßt werden um so die Qualität nachfolgender Befundungen zu verbessern. Für größtmögliche Betriebssystem-unabhängigkeit und Flexibilität wurde für die Realisierung Java 2 gewählt.

zu 7. Domänenunabhängigkeit:

Prinzipiell der ist das System unabhängig von einzelnen Domänen. Hiermit ist nicht nur gemeint, daß beliebige Befundkategorien gelernt werden können sondern auch andere Arten von Texten wie z.B. Formbriefe aus behördlichen Einrichtungen. Diese Systemeigenschaft konnte im übrigen in im Rahmen zahlreicher Testreihen bestätigt werden.

Das System wurde mit über 250 Befunden aus 3 Domänen in verschiedenen Testreihen, d. h. mit unterschiedlichen Regelkombinationen trainiert. Dabei hat sich gezeigt, daß die gewählten Regelmengen einen entscheidenden Einfluß auf die Benutzerfreundlichkeit der erzeugten Oberflächen haben. Prinzipiell konnte nachgewiesen werden, daß bei entsprechender Wahl der Systemparameter (insbesondere der Regelmenge) das vorgestellte System den Arzt bei der Befunderfassung unterstützen kann und durch seine genannten Eigenschaften, insbesondere der Domänenunabhängigkeit und Adaptivität vielen bisherigen Systemen überlegen ist.

Eine Demoversion findet sich unter:

http://homepage.ruhr-uni-bochum.de/kai.bullerdick/Diplomarbeit.html

13.2 Ausblick

Die Erstellung der Diplomarbeit und Realisierung von *mAGENTa* führte zu weitergehenden Überlegungen, wie die Thematik eines solchen Befundungsunterstützungssystems weitergeführt werden kann.

Hier sind vor allem zwei Gebiete für weitere Forschung zu sehen. Einerseits die Optimierung des Lernverfahrens, andererseits eine Verbesserung der Mensch/Agenten-Schnittstelle.

Im konkreten Fall könnte die reale Trennung der Lernkomponente vom TokenComponent-Interface zu einer Verschlankung des gesamten Systems führen. So müßte der Arzt den Befundungsvorgang nicht mehr an seinem PersonalComputer durchführen, sondern könnte mit einem Java-fähigen Handheldcomputer seinen Befund mobil erfassen. Anschließend werden per Infrarotanbindung an ein Intranet die Befunde zu einem Server übermittelt, der dann für viele verschiedene Benutzer den Lernprozeß übernimmt. Hierzu könnte auch auf das kooperative Optimieren des Lernergebnisses durch verschiedene Agenten Wert gelegt werden.

Eine Verschlankung des Lernvorganges würde auch durch Festlegung auf die Parameterkombination erreicht, die sich in der Diplomarbeit als günstig erwiesen hat. Auch wäre eine weitere Verbesserung der Lernalgorithmen z.B. durch parallele Problemlösungsstrategien denkbar. Da im Zuge des wissenschaftlichen Fortschritts gelernte Erkenntnisse revidiert werden könnten, sollte auch das Entfernen von bisher Gelerntem möglich sein.

Eine weitere Steigerung der Benutzerfreundlichkeit könnte durch die Integration einer Spracheingabe für das TCI erzielt werden. Der Arzt spricht die ersten Silben oder Worte, und der Satz wird gemäß der Struktur des DFA vervollständigt.

Weitere Textformatierungen, wie z.B. fett, kursiv sowie die Verwendung von verschiedenen Schriftgrößen und -arten, dienen einer übersichtlicheren Gestaltung und können dem Lernprozeß als Anhaltspunkt für einzelne Abschnitte dienen.

Die Anzeige des Graphen könnte durch eine eigene Ansicht pro MetaAbschnitt verbessert werden. Ein beliebige virtuelle Fenstergröße ließe es zu, mehr Elemente gleichzeitig einsehen zu können.

Da der Lernvorgang prinzipiell sprachunabhängig ist wäre es wünschenswert auch die Benutzeroberfläche für andere Sprachen (z.B. Englisch,...) zu lokalisieren.

Anhang: Voreinstellungen

Merging Parameter

Parameter	Voreinstellung	Beschreibung
BeliebigeAnzahl-NachfolgerAbfrage	false	Gibt an, ob statt der ursprünglichen Regel 3a von Schlimmer und Hermens die erweiterte Version verwendet werden soll
DirektesAnwenden-DerRegelnAbfrage	false	Gibt an, ob die eingestellten Mergingparameter direkt oder erst nach Aufruf einer entsprechenden Operation (z.B. Einlesen eines Befundes) auf den aktuellen Graph angewendet werden sollen
MetaAbschnitt-Abfrage	true	Gibt an, ob Abschnitte mit identischen Meta-Titeln innerhalb eines Befundes zusammengefaßt werden sollen
MetaStruktur	true	Gibt an, ob eine Meta-Strukturierung erwünscht ist, oder die Befundtexte als unstrukturierte Tokensequenzen betrachtet werden sollen
regel1Abfrage	true	Gibt an, ob im Rahmen des Mergingvorganges die Regel 1 getestet werden soll
regel2aAbfrage	true	Gibt an, ob im Rahmen des Mergingvorganges die Regel 2a getestet werden soll
MetaMerging-VarianteAbfrage	false	Gibt an, ob der Mergingvorgang auf Meta-Ebene präskriptiv (false) oder deskriptiv (true) erfolgen soll
regel2a-VorgaengerAbfrage	2	Gibt den k-Vorgängerwert der Regeln 2a an
regel2bAbfrage	true	Gibt an, ob im Rahmen des Mergingvorganges die Regel 2b getestet werden soll
regel2b-VorgaengerAbfrage	2	Gibt den k-Vorgängerwert der Regeln 2b an
regel3aAbfrage	false	Gibt an, ob im Rahmen des Mergingvorganges die Regel 3a getestet werden soll
regel3a-VorgaengerAbfrage	1	Gibt den k-Vorgängerwert der Regeln 3a an

Parameter	Voreinstellung	Beschreibung
regel3bAbfrage	false	Gibt an, ob im Rahmen des Mergingvorganges die Regel 3b getestet werden soll
regel3b-VorgaengerAbfrage	1	Gibt den k-Vorgängerwert der Regeln 3b an
Unvollstaendige-Metastruktur-Abfrage	false	Gibt an, ob bei der Eingabe der Befundtexte auch Teilmengen der aktuellen Meta-Abschnitte akzeptiert werden

TokenComponent-Interface Parameter

Parameter	Voreinstellung	Beschreibung
anzahlTokenChoice	5	die Anzahl von TokenButtons mit dem selben Vorgänger- und NachfolgerZustand, ab der eine Darstellung in einer TokenChoice-Auswahl erfolgt
anzeige-NachfolgerSymbol	true	gibt an, ob bei angezeigten TokenComponents ein Symbol für die Eigenschaft des NachfolgeZustandes (Meta, Start, Terminal, Zyklus) angezeigt werden soll
anzeige-Steuerzeichen	false	gibt an, ob bei angezeigten TokenComponents die Steuerzeichen (\t, \n, \r) angezeigt werden sollen (jeweils mit einem \ zur Maskierung)
mehrfache-Metatitel	true	gibt an, ob Metatitel im BefundFenster mehrfach angezeigt werden sollen
mehrfache-Metazustaende	true	gibt an, ob MetaZustände im TCI mehrfach besucht werden können, falls nicht soll beim nächsten weiter gemacht werden
protokoll	true	soll die Position der angeklickten TokenComponents protokolliert werden?
briefkopf	null	der Text, der vor dem erstellten Text steht
brieffuss	null	der Text, der hinter den erstellten Text steht
briefkopf-Dateiname	Briefkopf.txt	der voreingestellte Dateiname für den Briefkopf
brieffuss-Dateiname	Brieffuß.txt	der voreingestellte Dateiname für den Brieffuß
nachFrequenz	true	gibt an, ob bei Betätigung des Beenden-Konpfes die TokenComponent mit der höchsten Frequenz oder die an erster Stelle im TCI genommen werden soll

Wörterbuch Parameter

Parameter	Voreinstellung	Beschreibung
anzahlEintraege	50	die maximale Anzahl von Einträgen, die auf einer Wörterbuch -Seite angezeigt werden, bevor die weiteren Einträge auf der nächsten Seite angezeigt werden

Tokenizer Parameter

Parameter	Voreinstellung	Beschreibung
space	true	Das Leerzeichen wird als Trennzeichen interpretiert
tab	true	Der Tabulator wird als Trennzeichen interpretiert
newLine	true	Das "Neue Zeile"-Zeichen wird als Trennzeichen interpretiert
carriageReturn	true	Das Wagenvorlauf-Zeichen wird als Trennzeichen interpretiert
punkt	true	Der Punkt wird als Trennzeichen interpretiert
komma	true	Das Komma wird als Trennzeichen interpretiert
semikolon	true	Das Semikolon wird als Trennzeichen interpretiert
strich	true	Der Bindestrich wird als Trennzeichen interpretiert
alsTokens	true	Die gefundenen Trennzeichen werden als eigenständige Token interpretiert
tokenInZahl	false	Zahlen werden als das Token "xxx" interpretiert

DelimiterParameter

Parameter	Voreinstellung	Beschreibung
start	#	ein MetaTitel beginnt mit diesem Delimiter in einem Befund
end	##	ein MetaTitel endet mit diesem Delimiter in einem Befund

Farbschema Parameter

Parameter	Voreinstellung	Beschreibung
zustandColor	Color.orange	ein Zustand wird orange dargestellt
metaColor	Color.red.brighter()	ein MetaZustand wird hell rot dargestellt
terminalColor	Color.red	ein TerminalZustand wird rot dargestellt
tokenColor	Color(250, 220, 100)	ein Token wird als die Farbe mit den RGB-Werten 250, 220, 100 dargestellt
selectedColor	Color.cyan.darker()	ein selektierter Zustand oder Token wird dunkel cyan dargestellt
fixedColor	Color.white	ein fixierter Zustand oder Token wird weiß dargestellt
kanteVon-ZustandColor	Color.lightGray	eine Kante von einem Zustand zu einem Token wird hell grau dargestellt
kanteVon-TokenColor	Color.gray	eine Kante von einem Token zu einem Zustand wird grau dargestellt
stressColor	Color.darkGray	die Kantenbeschriftung mit dem Abstand zwischen Zustand und Kante in Pixeln wird dunkel grau dargestellt
textColor	Color.black	die Beschriftung wird schwarz dargestellt

Netzwerk Parameter

Parameter	Voreinstellung	Beschreibung
connection	false	existiert eine Verbindung zum Netzwerk
context	null	der Kontext des Applets
codeBase	null	Die Basis-Adresse des HTML-Codes

Sortierungs Parameter

Parameter	Voreinstellung	Beschreibung
sortierung	true	sollen die Token sortiert werden?

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Adlassnig 1993] Adlassnig, K.-P.; **Wissensbasierte Entscheidungsunterstützung in der Medizin**;1993

[Ahonen 1996] Ahonen, Helena; **Generating grammars for structured documents using grammatical inference methods**; Dissertation, Universität Helsinki; 1996

[Altenkrüger et al. 1992] Altenkrüger, Doris/ Büttner, Winfried; **Wissensbasierte Systeme**; Vieweg; 1992

[Angluin 1983] Angluin, D.; **Inference of Reversible Languages**; in journal of the association of computing machinery; Vol.29; Nr.3; S.741-765; 1983

[Angluin et al. 1983] Angluin, D./Smith, C.H.; **Inductive Inference: Theory and Methods** ; in ACM Computing Surveys; Vol.15; Nr.3; S.237-269; 1983

[Beran 1984] Beran, Georg; **Textgenerierung zu CT-Untersuchungen über befundorientierte Entscheidungslogik**; Diplomarbeit, Fachbereich Informatik, Universität Dortmund; 1984

[Bernauer 1991] Bernauer, J.; **Conceptual graphs as an operational model for descriptive findings**; in SCAMC 1991; S.214-218; 1991

[Berner et al. 1999] Berner, E.S.(Hrsg); **Clinical Decision Support Systems – Theory and Practice**; Springer; 1999

[Berwick et al. 1987] Berwick, Robert/Pilato, Sam; **Learning Syntax by Automata Induction**; in Machine Learning; Nr.2; S.9-38; 1987

[Boegl et al. 1995] Boegl, K. et al.; **Neue Ansätze zur computerassistierten Diagnose rheumatologischer Erkrankungen**; in Radiologie; Nr.35; S.604-610; 1995

[Burgener et al. 1997] Burgener, F. A./ Kormano, Martti; **Differentialdiagnose in der Computertomographie**; Thieme Verlag; 1997

[Caglayan et al. 1997] Caglayan, Alper K. / Harrison, Colin G.; **Agent Sourcebook;** John Wiley & Sons; 1997

[Comer 1999] Comer, Douglas E.; **Computer Networks and Internets 2nd ed.;** Prentice Hall; 1999

[Dupont et al. 1994] Dupont, P./Miclet, L./Vidal, E.; **What is the search space of the regular inference ?**; in Proceedings of the 2nd International Colloquium on Grammatical Inference (ICGI94); S.25-37; 1994

[Dupont 1996] Dupont, P.; **Incremental regular inference**; in Grammatical Inference: Learning Syntax from Sentences, (ICGI96); Lecture Notes in Artificial Intelligence (1147); S.222-237; Springer; 1996

[Dupont 1997] Dupont, P.; **Grammatical Inference: Formal and Heuristic Methods**; Carnegie Mellon University; Pittsburg PA; 1997

[Finch 1993] Finch, Steven P.; **Finding Structure in Language**; Dissertation, Centre of Cognitive Science, Universität Edinburgh; 1993

[Flanagan 1998] Flanagan, David; **Java in a Nutshell 2nd Edition**; O'Reilly & Associates; 1998

[Fu et al. 1975] Fu, K.S./Booth, T.L.; **Grammatical inference : Introduction and survey (part I)**; in IEEE Transactions on Systems, Man and Cybernetics; Vol.5; S.95-111; 1975

[Goan et al. 1996] Goan, T./Benson, N./Etzioni, O.; **A Grammar Inference Algorithm for the World Wide Web**; in Proceedings of the 1996 AAAI Spring Symposium on Machine Learning in Information Access (MLIA’96); Stanford; CA; AAAI Press; 1996

[Gold 1967] Gold, M.; **Language Identification in the Limit**; in Information and Control; Vol.10; S.447-474; 1967

[Gold 1978] Gold, M.; **Complexity of Automaton Identification from given Data**; in Information and Control; Vol.37; S.302-320; 1978

[Hellbig 1991] Hellbig, Hermann; **Künstliche Intelligenz und automatische Wissensverarbeitung**; Verlag Technik; 1991

[Heyder et al. 1988] Heyder, N./Lederer, P./Schmidt, H./Grassme, U.; **Der sonographische Befund aus dem Computer**; in Deutsches Ärzteblatt; Nr.85; S.443-448; 1988

[Hopcroft et al. 1988] Hopcroft, J./Ullman, J.; **Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie**; 1.Auflage; Addison-Wesley; 1988

[Krüger 1999] Krüger, Guido; **GoTo Java 2**; Addison-Wesley-Longman; 1999

[Kuhn 1994] Kuhn, K. et al.; **Strukturierte Befunderstellung für die Oberbauchsonographie, Erfahrungen und Perspektiven**; in Ultraschall Klinische Praxis; Nr.9; S.43-46; 1994

[Kuhn 1996] Kuhn, K.; **Die elektronische Befund- und Bildokumentation in der Sonographie**; in Radiologe; Nr.36; S.47-50; 1996

[Langenscheidt 1991] Langenscheidt ; **Taschenwörterbuch Englisch**; 20. Auflage; Langenscheidt; 1991

[Leiner et al. 1998] Leiner, F./Gaus, W./Haux, R.; **Medizinische Dokumentation**; 2.Auflage; Schattauer; 1998

[Lipinski 1999] Lipinski, H.-G.; **Einführung in die medizin-technische Informatik**; Oldenbourg; 1999

[Lissner et al. 1992] Lissner, J./Fink, U.; **Radiologie I**; 3.Auflage; Ferdinand Enke Verlag; 1992

[Lüth 1998] Lüth, Tim; **Technische Multi-Agenten-Systeme**; Carl Hanser Verlag; 1998

[Meyer 1998] Meyer, André; **JFC 1.1 mit Java Swing 1.0**; Addison-Wesley; 1998

[Mitchell 1997] Mitchell, Tom; **Machine Learning**; McGraw Hill;1997

[Muggleton 1990] Muggleton, S.; **Inductive Acquisition of Expert Knowledge**; Addison Wesley; Reading, MA; 1990

[Morik 1997] Morik, K.; **Einführung in die künstliche Intelligenz**; 5.Auflage; Lehrstuhl VIII; Fachbereich Informatik; Universität Dortmund; 1997

[Morik 1999] Morik, K.; **Maschinelles Lernen**; Lehrstuhl VIII; Fachbereich Informatik; Universität Dortmund; 1999

[Oncina et al. 1992] Oncina, J./Gracia, P.; **Inferring regular languages in polynomial update time**; in Pattern Recognition and image analysis; S.49-61; 1992

[Parekh et al. 1998] Parekh, Rajesh/Honovar, Vasant; **Grammar Inference, Automata Induction and Language Acquisition**; in Handbook of Natural Language Processing; New York; 1998

[PG 281 1996] Projektgruppe 281; **Moses, ein medizinisch orientiertes**

Sprachevaluationssystem; Abschlußbericht, Fachbereich Informatik, Lehrstuhl I; Universität Dortmund; 1996

[Pitt 1989] Pitt, L./Warmuth, M.K.; **Inductive Inference, dfas and computational complexity**; in Analogical and Inductive Inference; Lecture Notes in Artificial Intelligence 397; S.18-44; Springer; 1989

[Pschyrembel 1994] Pschyrembel; **Klinisches Wörterbuch**; 257. Auflage; de Gruyter; 1994

[Schlimmer et al. 1993] Schlimmer, Jeffrey C./Hermens, Leonard A.; **Software Agents – Completing Patterns and constructing user interfaces**; in Journal of artificial intelligence research, Nr.1 ; S. 61-89; 1993

[Schröder 1993] Schröder, M.; **Sprachverarbeitung in der Medizin – Anwendungen und Methoden**; in KI; Nr.2; S.50-55; 1993

[Seelos et al. 1997] Seelos, Hans-Jürgen (Hrsg.)**; Medizinische Informatik, Biometrie und Epidemiologie**; de Gruyter; 1997

[Stowasser 1980] Stowasser, Josef M.**; Der kleine Stowasser: Lat.-dt. Schulwörterbuch**; G. Freytag; 1980

[Thurn et al. 1998] Thurn, P./Bücheler, E./Lackner, K.J./Thelen, M.**; Einführung in die radiologische Diagnostik**; 10.Auflage; Thieme Verlag; 1998

[Trakhtenbrot et al. 1973] Trakhtenbrot, B./Barzdin, Y.; **Finite Automata: Behavior and Synthesis**; North Holland Publishing Company; Amsterdam; 1973

[Wegener 1996] Wegener, Ingo; **Effiziente Algorithmen für grundlegende Funktionen**; 2. Auflage; Teubner; 1996

Abkürzungsverzeichnis

API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BWS	Brustwirbelsäule
CT	Computertomographie
DFA	Deterministic Finite Automat (Deterministischer Finiter Automat)
DFS	Depth First Search
EBT	Eletronic-Beam-Tomography
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
HWS	Halswirbelsäule
IP	Internet Protocol
JAR	Java Archive
JDK	Java Development Kit
JFC	Java Foundation Classes
JRE	Java Runtime Environment
LWS	Lendenwirbelsäule
MRT	Magnetresonanztomographie
Pixel	Picture Element (Bildpunkt)
RGB-Modell	Rot, Grün, Blau-Modell
TCI	Token Component Interface
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
URL	Uniform Resource Locator

Index

Erklärung

Name, Vorname ....

Hiermit erkläre(n) ich (wir), daß ich (wir) die Diplomarbeit mit dem Titel:

....

....

....

....

....

selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt habe(n).

Dortmund, den ....

Unterschrift